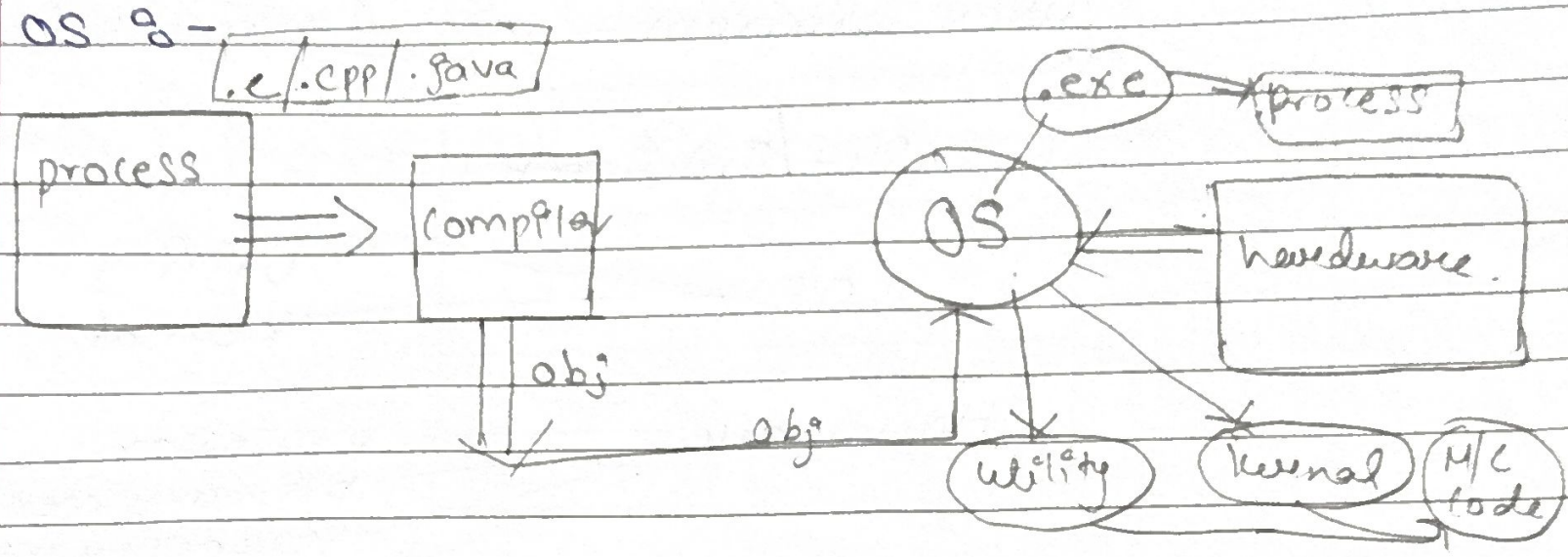
PAGE NO.:
DATE: / /

OS/02/25 Unit-1 process Management

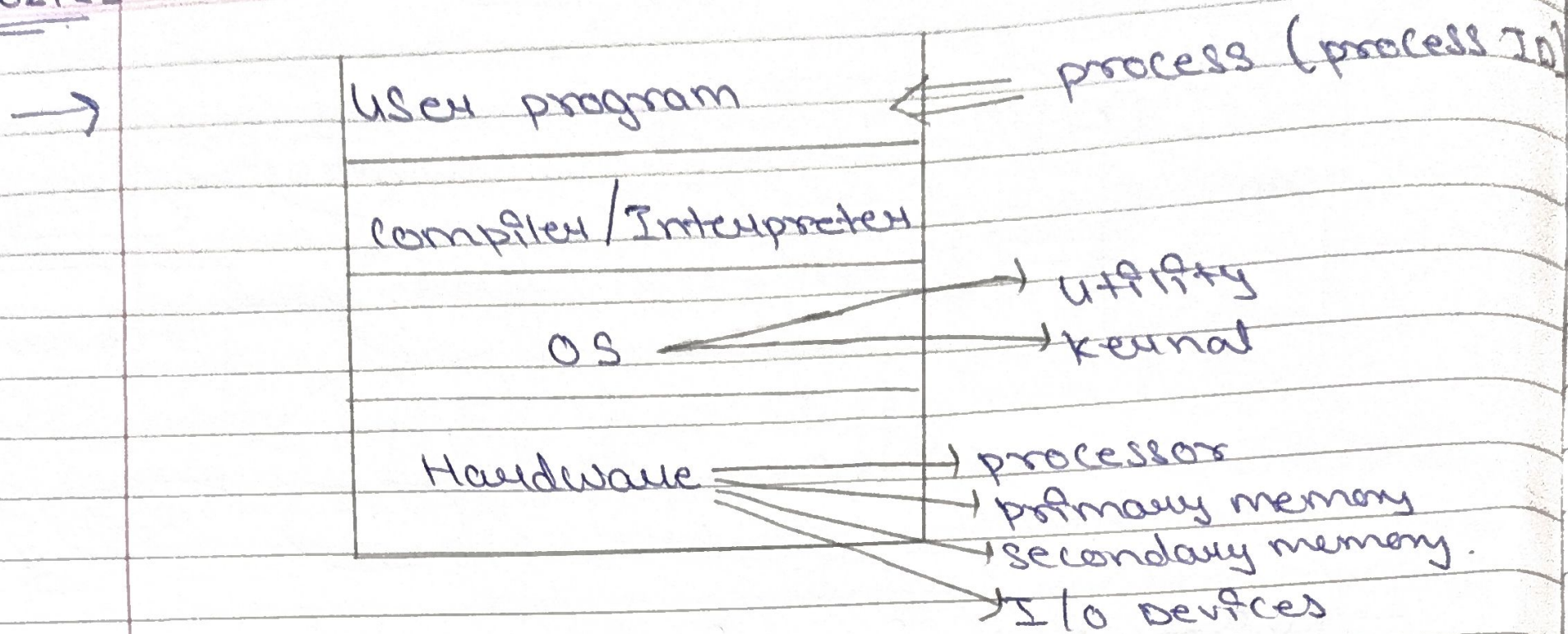
→ OS :-



– Major goal of OS is :-

- To create a process for an application (which will be written by programmer and compiled by compiler).
- This newly created process is executed by OS.
- The resources (hardware) demanded by process, is provided by OS.
- OS can create a child process from previous process
- After successful execution of process (after getting output of the program / Application), process will be terminated by OS.
- Multiple processes are executed concurrently / simultaneously within the system and multiple processes are demanded resources from the OS.
- OS has the major challenge to distribute the limited resources to multiple processes in such a way, all processes can complete their execution successfully within minimum time.

08/02/25



Imp. [2 or 5 M] → Justify :- OS is a Resource Manager.

- Limited resource will be supplied to multiple processes simultenously to complete the execution of multiple processes.
- This parallel process execution will be done by O.S
  That's why, the O.S is called Resource Manager.

10/02/25

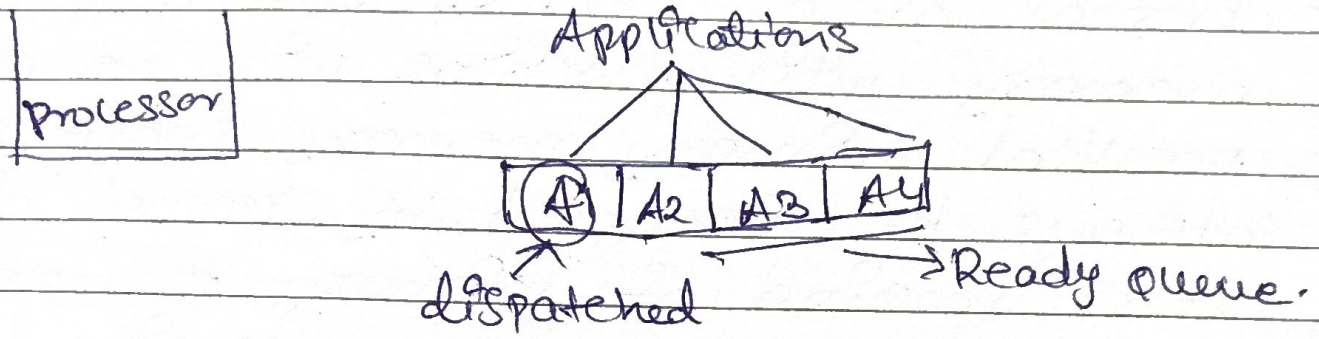
→ Demanding Resources :-

- programmer / user is demanding the hardware resources in the form of program.
- The resources is provided or supplied by OS for execution of program.
- So, user / programmer's or client OS.

- The task of OS is satisfied the programmer to generate the correct output using

minimized resources.

- Dispatcher is dispatching the applications to the processor for a execution.
- Dispatcher is called as a processor scheduler.



→ Memory Manager :-

- Memory manager part is OS, who is managing the primary memory allocation.
- primary memory is demanding by applica-tion.
- local variables ~~within~~ ~~with in~~ which are declare within the function / methods are allocated with in primary memory by memory manager.
- After returning from function / methods, all local variables are ~~alle~~ deallocated by memory manager.
- The global variable are C application or instance variable of the object are allocated within memory by memory manager.
- After compilation application or after deletion of the object, all global and instance variable of object are allocated by memory manager / garbage.

11/02/25

→ Kernal of OS :-

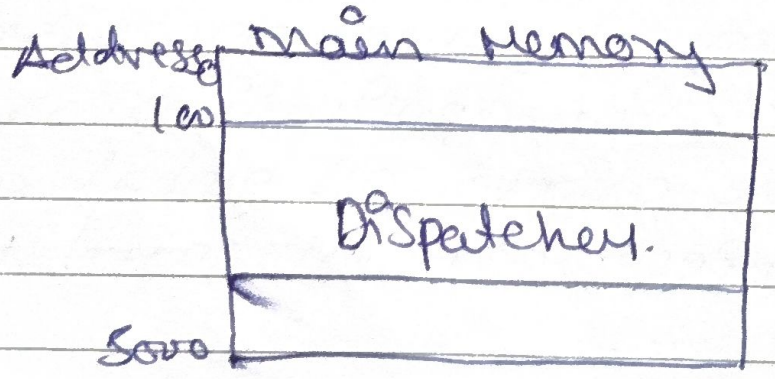
- When machine will be started then kernal of OS will be loaded within primary memory.

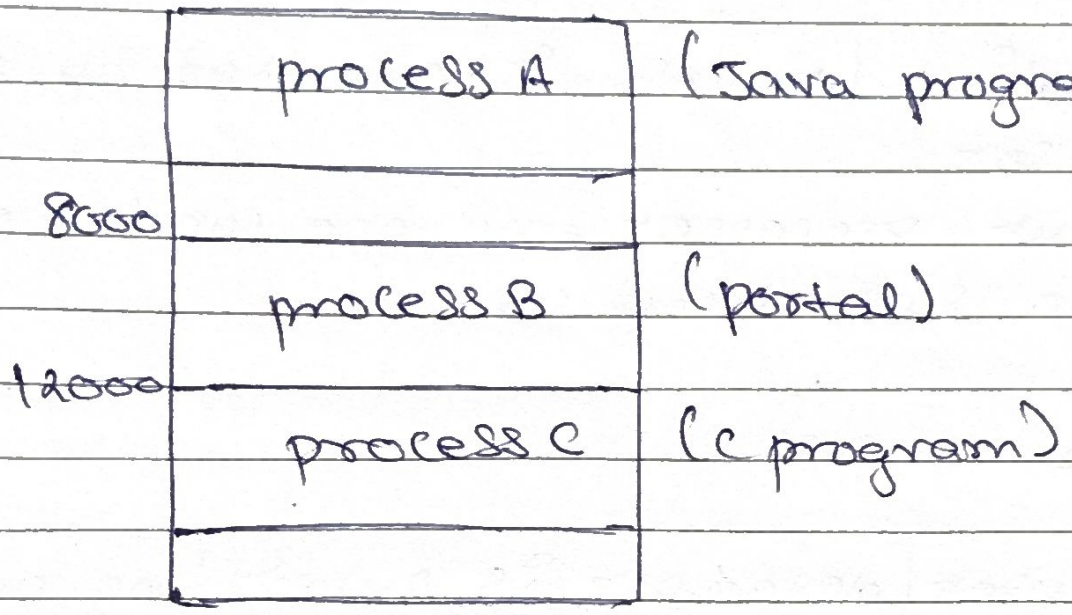
- kernal of OS resides within primary memory until machine will be shut down.
- partial primary memory space will always be occupied by kernal of OS, rest portion of primary memory is available for user application / process.

- At the time of the execution of any application / program / input data will be swapped in loaded within primary memory from hard disk.

- After successful execution of application, program will be swapped out from primary memory to secondary memory, This swapped out and in task will be managed by memory manager of OS.

→ Multiprogramming :-





Q. Explain multiprogramming or multitasking using uniprocessor.

Ans More than 1 programs are submitted to the system for execution.

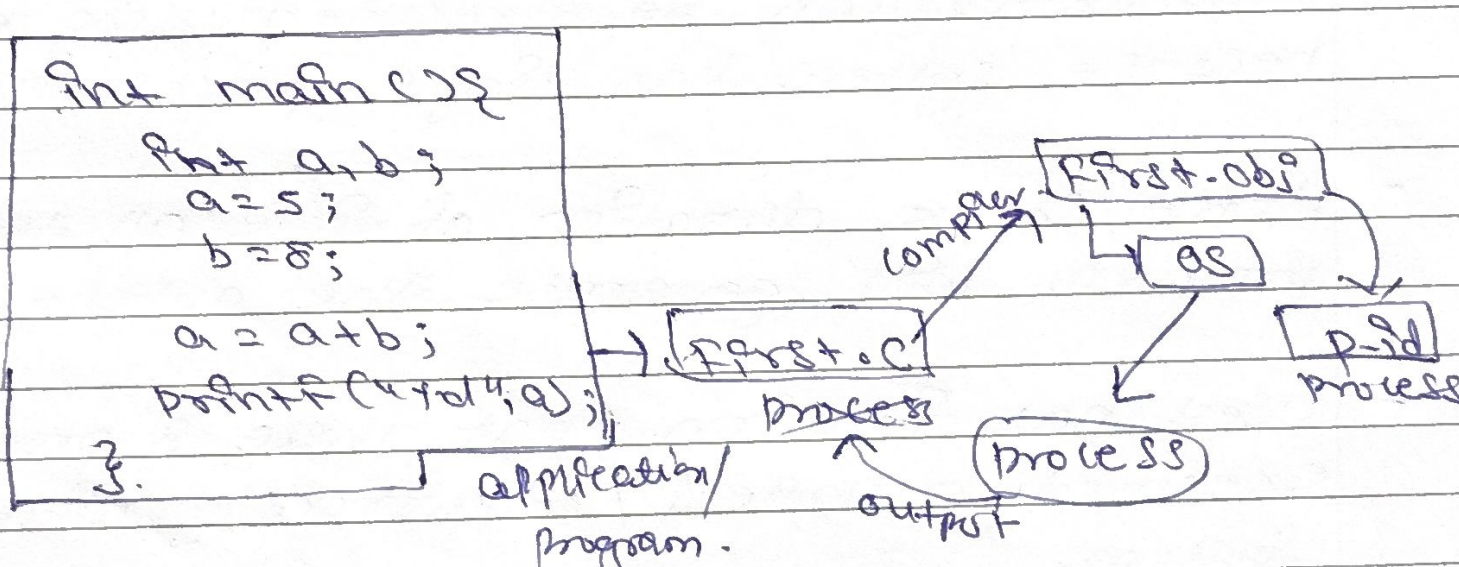
- 3 programs are loaded within primary memory from secondary memory.
- dispatcher will slice the processor time within 3 programs.
- processor will be switched between 1 program to other program based on scheduling of dispatcher.
- At a single point of time, only 1 programs instructes can be executed using processor because system has single processor.
- After some duration of time all these program will generate the output.
- user has intution that these 3 program are executed parallely / concurrently within system.

12/02/25

- Application / process is executed by OS using processor.
- Without processor, no one can be executed within system by OS.

20/2/25 chp-3 process

- When user / developer is submitting any applic-ation for execution, a process is created by OS.
- For a process, a process identifier (processId) will be generated by OS.
- Within system, each process has its parent proces
- process exists within system (with in primary memory) until output will be generated by process.
- During the life span (life cycle) of the process in (between creation and termination of the process); each instruction execution of the process will be controlled by OS.
- The resources (primary memory and proces-sor time) which are demanded by process, is supplied by OS.



- Definition of process :-
process is the instance of an application (First.c) in execution.

→ process Image :-

program First.c

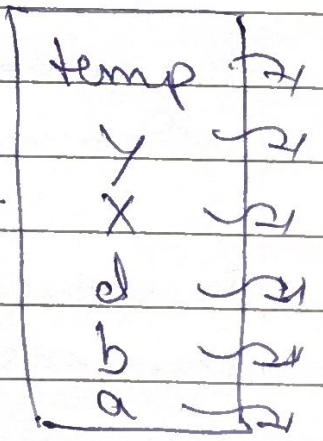
- text - instruction code of program written by programmer
- Stack - temporary data Function parameter local variable. } Input data, output data
- heap - Dynamically allocated data, variable.
- data - global variable.

heap is assign by OS process.

```
text
int main()
{
    int a,b;
    int d;
    a=5;
    b=6;
    a=a+b;
    d=F(a,b);
}
```

```
int F(int x, int y)
{
    int temp;
    temp = x+y;
    return temp
}
```

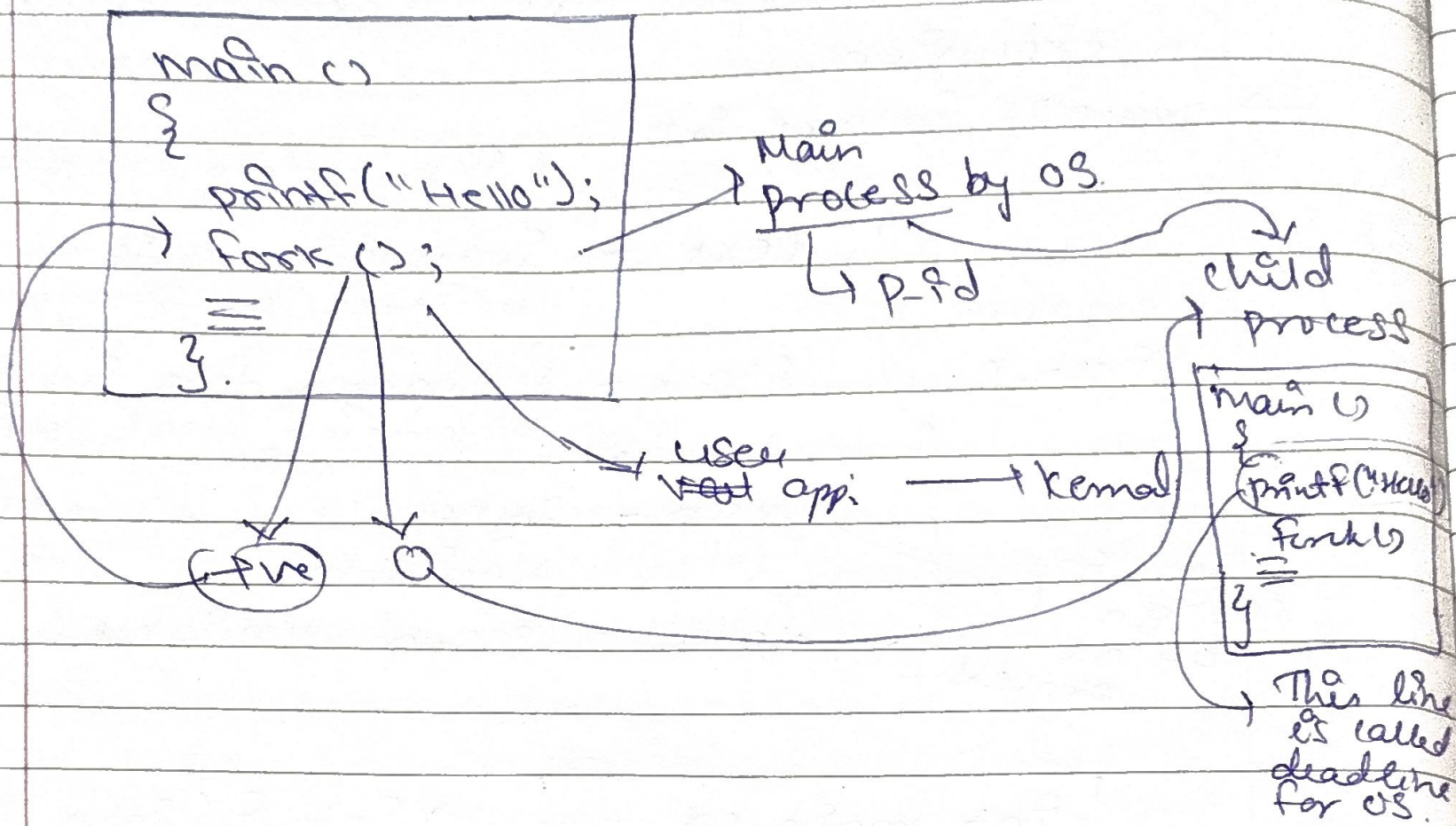
Stack →



03/03/25

chapter- 3 process

→



- Return value of fork within parent process/ main process is equal to the process id of child process.

- Fork function is returning positive Integer value to parent process, that returned value will be assigned as a process id of child process.

A process is Associated with program code/ process Image data and different attributes of the process (PCB - process control Block).

After creation of a process PCB will be created for a process.

- PCB of a process exist with in system until the process will be terminated. After execution of it's program code.

- PCB of a process is created by OS, for Managing a process/execution of a process with in system, PCB of that process is Refered by OS.

05/03/25

- PCB is the structure in which detailed [2.5M] information of a process is stored.

- For an individual process, individual PCB's is created by OS.

- PCB is created by OS, is handled by OS and deleted by OS.

- First attribute of PCB is process_Identifier (p-id, parent p-id, user id).

- Second attribute of PCB is processor state information.

- Third attribute of PCB Data structure, Memory mangement, I/O devices, Scheduling and accounting.

06/03/25

Chpt - 3

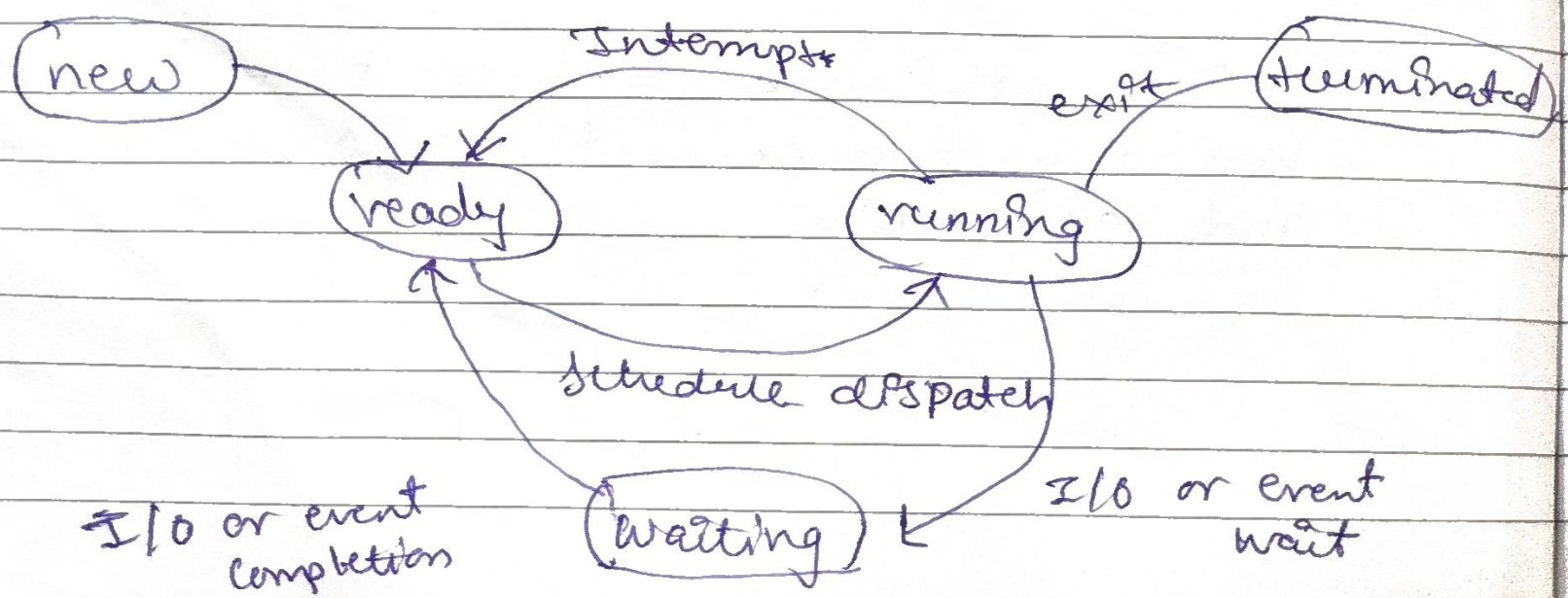
→ process State Transition :-

When the program get executed the process will created.
Then the process is created it will be in new state. (more than 1 process can be there).
Then the process will be in ready state when process get its processor.
Then it will passes to running state at a single moment single process will be assigned at running state because the processor is single it can process only one process.

as will decide that which process will go to the running state and meets to processor

When the time slice is over and some Interruption occurs then os will send process from running to ready state.

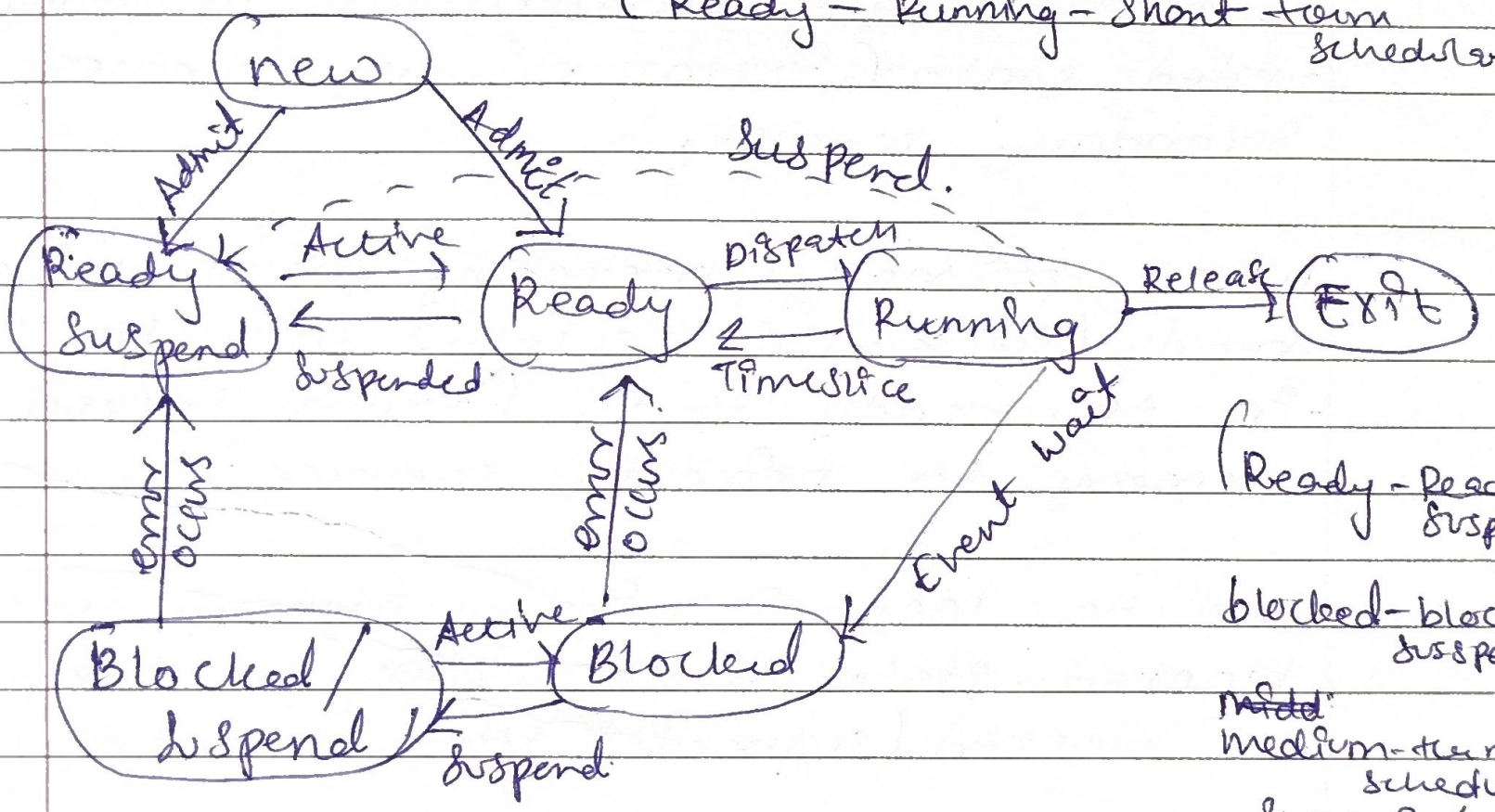
Diagram :-



When there is a lack of resource in the process then the process will be jumping to waiting state. And when the OS will notify that now you can move to ready state.

When there is lack of resource and also not having space in primary memory then the process will be in waiting to suspended state and the O.S will notify that now you can move to ready state.

Diagram :- CPU (New - Ready - long term scheduler)
for (Ready - Running - short term scheduler)



08/03/25

- process will go to suspended state due to lack of space in primary memory.

- The process which are in ~~ready~~/Suspended state (ready/suspended or Block/Suspended) that

PAGE NO.:
DATE: / /

Indicates, all those processes are with in Secondary memory.

- The processes which are in ready or Block state that indicates processes are in primary memory.

- Suspended state of a process means, the processes are in Secondary / Virtual Memory.

- Due to lack of primary memory space, ready state processes or new state processes are swapped-out to ready / Suspended state (from primary memory to Secondary memory).

- When primary memory will available, ready (suspended processes will be swapped-in to ready state (from Secondary memory to primary memory).

- Due to lack of primary memory space blocked-state process will be swapped-out to blocked / suspended state (secondary memory)

- Due to lack of resources and there is space in primary memory then the process will go from block / suspended to blocked (waiting) state.

→ process Switching :-

- processor will be switched from one process to another process.

- Running state process will be moved from running state to Exit / Ready / Blocked / Ready Suspended State.

- One Ready state process will be dispatched only one from ready state to running state. So, State of the process will be switched from the running to ready (Blocked / Exit) and ready to running.

10/03/25

Imp Q What are the functionality of OS ?

Q What is the difference between compiler/ Interpreter and OS.

Q what is System call ? Explain with linux c program.

Q What is process ? Who will create the process and when will the process created.

Q What is process Image ?

Q Explain Stack area, heap area and data

area of primary memory space. And also Explain which type of data are stored in which area of primary memory space.

[7M] Q What is PCB ? When PCB is created & by whom it is created.

Q Explain processor state ~~transf~~ information

Q Define process Identifier (p_id, pp_id & User_id) (getp_id, getpp_id).

Q Explain process state Transition Diagram.

Q Explain Individual state Transition with reason.

Q Write down the Reasons for process crea-tion and termination.

Q Explain the suspended state (Ready/Suspend and Block/Suspend).

Q What is process switching (with diagram).

IMP Q Mode switching (with fork(), with code).

Ans When user program is executing within using processor means processor is in User Mode.

If any system call (fork()) is invoked by user program, control of the execution will be switched from user application to O.S.

At the time of execution of system function, the processor is in kernal mode. (higher privilage).

After successfull execution of system call, control will be switched from system call to user application.

Exp?

Q Difference between process switching and Mode switching.

Q Explain process spawing.
For a pevent no. of children is created is called process spawing.

- Thread is called light weight process which occupy lesser space.

→ Inter process communication (IPC) :-
(SHM gate Hard) A process is communicating to other process through shared memory and message passed.

11/03/25

→ Thread :-
process is called heavy weight process whereas thread is a light weight.

→ parallelism :-
More than one task are executing simultaneously. using multiprocessor.

→ Sequencial :-

```
for(i=0; i<10; i++)
        sum = sum + a[i];
```

```
p_id = fork();
if (p_id == 0) {
        for (i=0; i<25; i++)
            sum_child = sumchild + a[i];
}
else {
        for (i=25; i<49; i++)
            sum_parent = sumparent + a[i]
}
```

→ ~~Concurrently~~ Individual task will be assign to individual process / thread.

So, Multiple process for multiple thread are executing simultenously are using multiple processor.

→ Concurrency :-
Multiple processes/multithreads are executing simultaneously using uni/single processor.

→ Data parallelism :-
When there is a large amount of data set/array element dataset will be splitted into number of subset individual subset will be handled by individual thread.
So multiple threads are executing parallelly / concurrency for processing the subset of the dataset.

Dataset : Array of Integers of size 50.
Subset of Data : Array/Subarray 1 0-24
25-49 Subarray 2
Task is computation of sum.

→ Task parallelism.
A single task will be divided into multiple task. Individual subtask will be assign to individual thread and then multiple thread are executing their multiple subtask parallely / concurrently.

24/03/25 Chp-5 process Synchronization. (Unit-2)

→ process Synchronization, concurrency :-

- principles of concurrency
- Definition of concurrency, Shared memory, local memory
- problem generated due to concurrency.
  - Contention
  - Race condition
- Solution of the problem
  - Introduction of Semaphore.
  - Functionalities of Semaphores.
  - Implementation of critical section using Semaphores
  - Different algorithms related to critical section.

→ Concurrency :-
When more than one process / thread are executing simultaneously using uniprocessor, it is called concurrency.

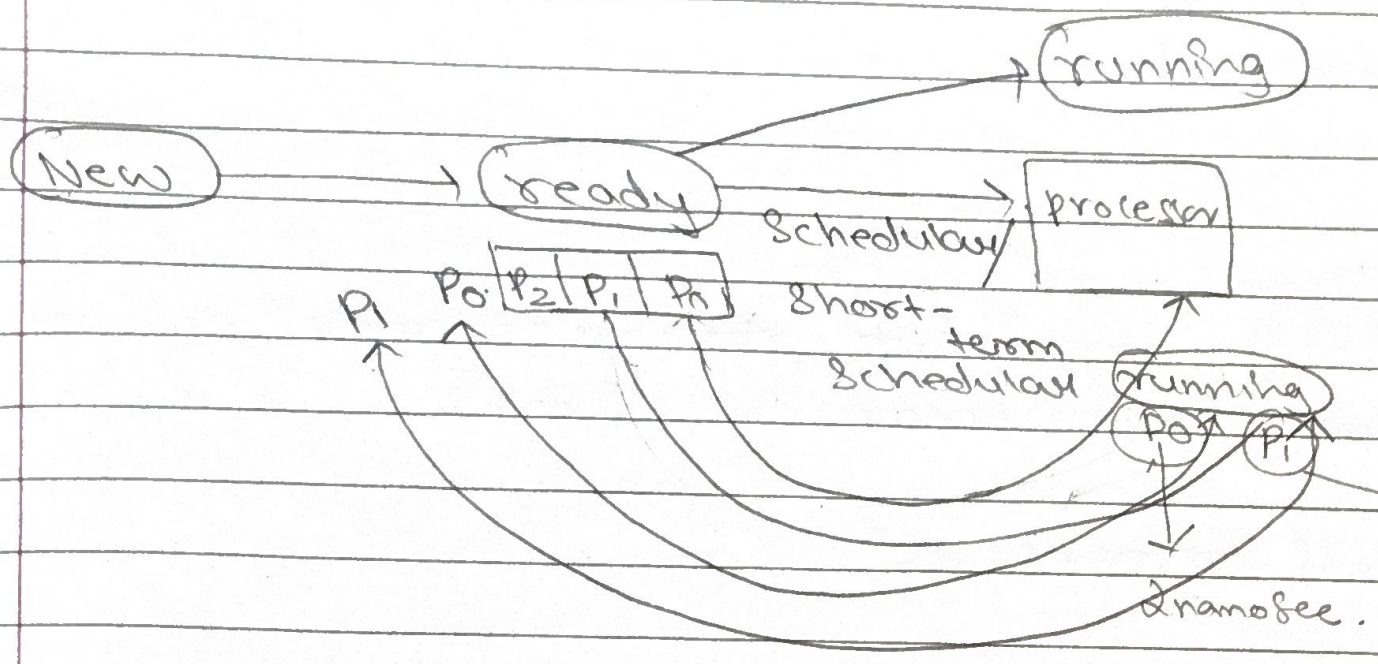
- When more than one process / thread are executing simultaneously using multi-processor, is called parallelism.

- In concurrency and parallelism, same technique are implemented by OS and programmer.

- When we are having same shared resource then the contention problem is occured.

→ principles of concurrency :-

- when more than one process are executing simultaneously, Scheduler of OS is using time slicing / Inter leaving technique.



- Short-term Scheduler assigns / gives the time / selice to individual ready queue process, one process from ready queue will be dispatched to processor / running state, after completion of given time slice running state process will release / leave the processor / running state and will move to ready state and immediately one of the ready process will be dispatched to processor / running state.
- This flow of the switching of the process within processor is called Inter leaving technique.

25/03/25.

→ Contention :-
When more than one process are accessing / Modifying single shared resource simultaneously / concurrently using uni-processor / multiprocessor, there is a high chance to generate incorrect output. This problem is called contention.

26/03/25

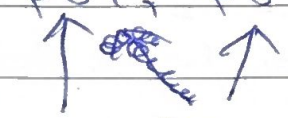
- Single processor multiprogramming (processes are interleaved in time)

pri(processor)

p1, p2, p3 (Interleaved/Ing)

- Multiple processor multiprogramming (processes are overlapped)

pr1, pr2 (processor)



p1, p2, p3, p4, p5... processes
(Interleaving vs overlapping)

- In multi processor system (pr1, pr2), p1 process is dispatched to pr1 processor and at the same time p2 process is dispatched to pr2 processor. So, p1 and p2 process are executing (concurrently / parallelly. It is actual concurrency/

parallelism. This technique is called overlapping.

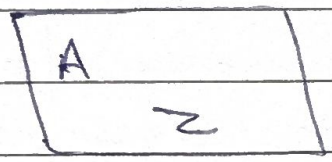
- Uniprocessor system actual concurrency is not ~~implemented~~ feasible.

- In uni processor system Interleaving technique is provide by scheduler.

- Contention - The problem generated due to concurrency.
(Q) contention
~~when more than~~

Eg:-

Process P1 Process P2



chin (shared resource)

1. chin = getchar();
(A)

2. chin = getchar();
(z)

3. chout = chin;

4. putchar(chout);
chin = 'z'

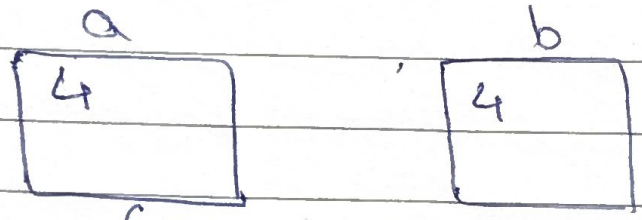
5. chout = chin;

6. putchar(chout);
chin = 'z' (error)

- Race condition :-

```
p1(a1) // p1 process
{
a = a+1;
b = b+1;
}
```

```
p2(a2)
{
b = 2*b;
a = 2*a;
}
```



(Shared resource)
/variable.

- p1 process is called/executing p1 function.
- p2 process is executing p2 function,
- but p1 and p2 both process are accessing and modifying same shared resource a and b.

28/03/25

- a=b=4

Q Explain ~~race contention and contention~~.
Explain contention and Race condition.
How this problems are generated.

Q How can we solve this problem. (Implementation of Mutual exclusion / critical section using semaphore)

→

p1()
```
a = a+1
b = b+1
```

p2()
```
b = 2*b
a = 2*a
```

29/03/25

→ Critical section :-
When more than one processes are accessing and modifying the shared resource, the instructions related with shared resource should be put into critical section.

Critical section is controlled by semaphore / Monitor.

When there is no process within critical section, only one process can enter into the critical section.

After entering to the critical section, the enterant process immediately locks the critical section and then enterant process will access the shared resource.

* No other process can enter into the critical section until the enerant process will exit from critical section.

When the enerant process will exit from the C.S, it will send the signal to awaited process who is trying to enter to critical section.

Now, next process will be allowed to enter into the critical section.

— A semaphore which is controlling the critical section, that semaphore is shared by the processes who are executing single the critical section associated with common shared resource, though the instruction of the critical sections are different.

p1()
```
a = a+1
b = b+1
```

p2()
```
b = 2*b
a = 2*a
```