[1] Utilization

Precentage of time a recource is in use

[2] Throughput

To measure the output of the system

rate → 3 processes / min

[3] Turn around time:

Time taken between submitting a process until it's completion

[4] Response time:

Time taken between submitting a process and getting the first response

Example

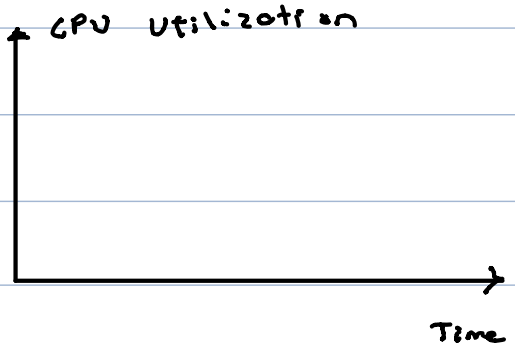
	Duration	CPU	Memory	Disk	Terminal	Printer
A	5 min	70	50MB	N	N	N
B	15 min	10	100MB	N	Y	N
C	10 min	10	75MB	Y	N	Y

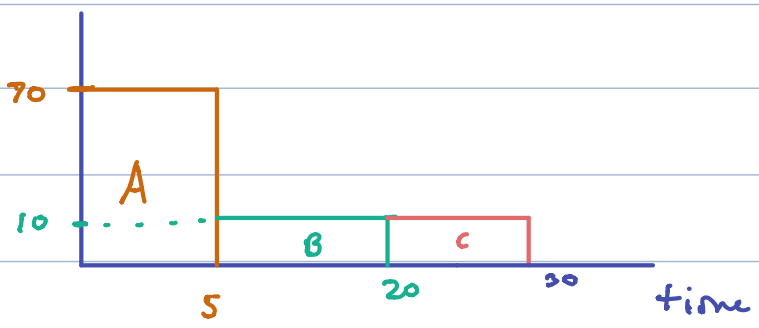
Total memory = 250 MB

- Case A

Uniprogramming (Serial)

1. Compute average CPU Utilization





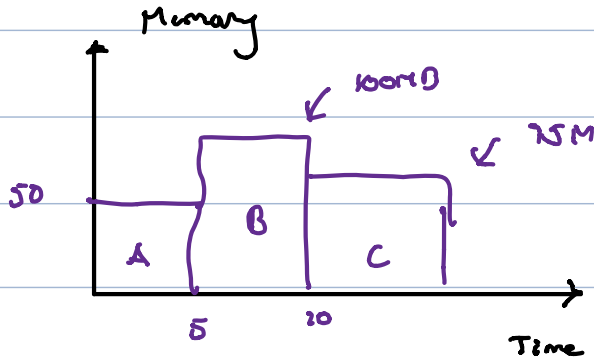
$$\frac{5 \times 70 + 10 \times 15 + 20 \times 10}{30}$$

↓ leads to

20%.

- those values lead to percentages

2. Average Memory Utilization



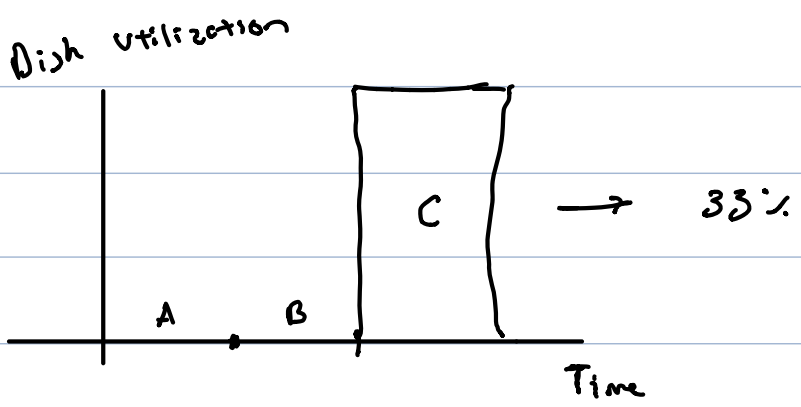
$$\frac{50 \times 5 + 100 \times 15 + 75 \times 10}{30 \times 250}$$

↑ the total

then multiply by 100 for percentage

= 33%.

3. Average disk Utilization



4. Throughput

6 process / hr

5. Average turn around time

= 5 + 20 + 30

A B C

All arrived at time ∅

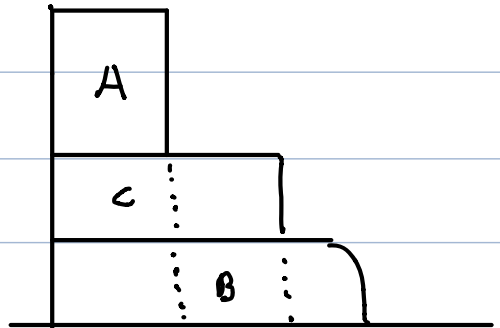
Divides it by 3

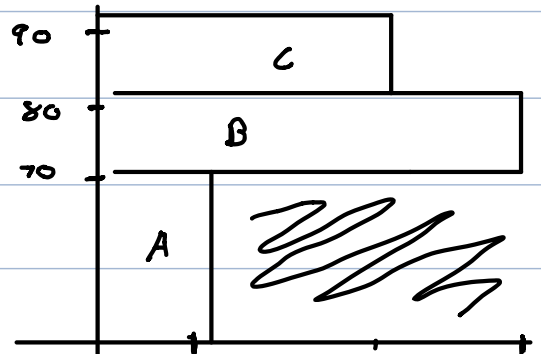
= 18 minutes

- Case B

Multiprogramming

1. Average CPU Utilization

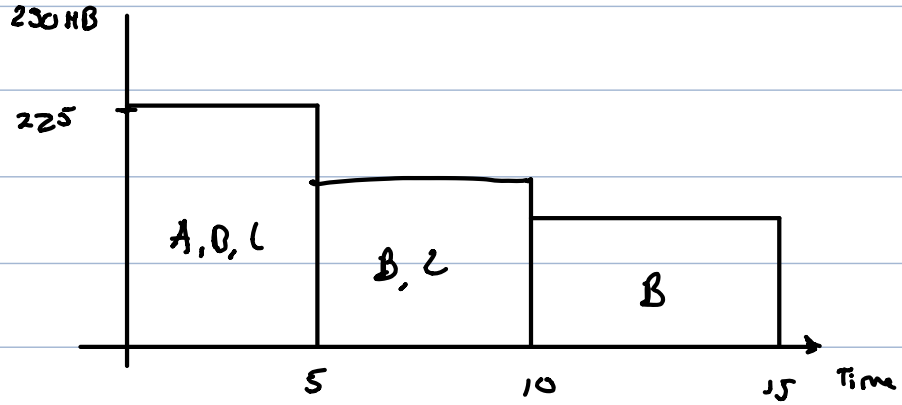




5 10 15 5 10 15

↑ Preferred Diagram

2. Average memory utilization



5 second increments

$$\frac{(225 \cdot 5) + (175 \cdot 5) + (100 \cdot 5)}{15 \times 250}$$

time × memory

total

3. Throughput = 12 processes / hr

Avg turn around time

$$\frac{5 + 10 + 15}{3} \rightarrow 10 \text{ min}$$

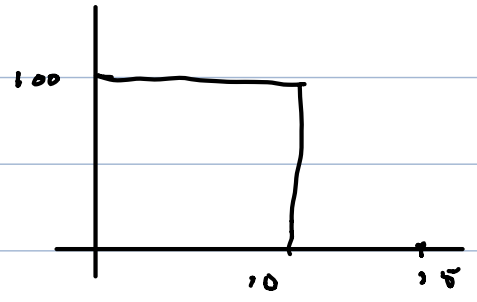
b/c

ABC done in 15

$15 \cdot 4 = 60$

$3 \cdot 4 = 12$

Multiprogramming Dash Utilization



$= \frac{10}{15} \cdot 100$

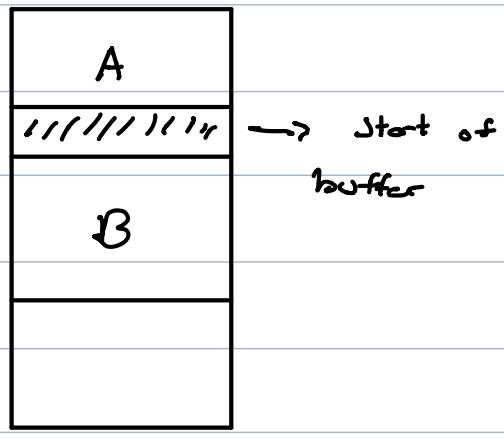
Interprocess Communication

Need to all process to comunicate

why?

1. Sharing Data
   - Modulcriz the applicction design
   - speedup (only on multi processor systems)

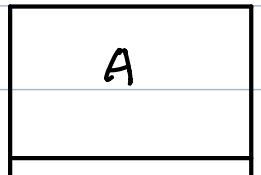
2. Shard memory
   - Message passing



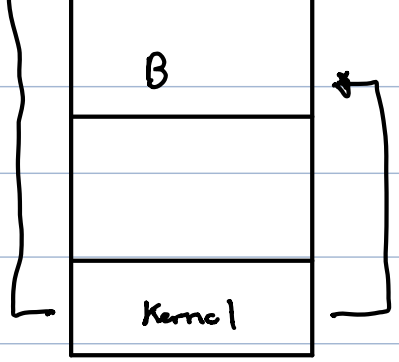
Establish a shared region in the memory. that resides in the addresses spece of the process creating the shared memory

- Other processes attech to this region
  - -> Need to remove OS restrictions for protection
  - -> Access Synchronization

Message Passing



- Every message requires the kernal
  - -> Send (P, , msg) } symmetric
  - -> Recieve (q, , msg) }



only the sender names the recipient

- Indirect Communication

Messages sent / recieved through mail boxes (ports)

OS

-> Create mailbox
-> Delete mailbox
-> send message
-> recd message

Non blocking → you don't wait
blocking → wait till you get something

Blocking Send
Non .... send
blocking recieve
" " .... recieve

} Different functionalities