Operating System.

Operating System.

- A program that acts an intermediary between a users of a computers and the computers hardware.

bridge

System Software

- System Software refers to the operating system and all the utility programs that manages computers resources at a low level.
- System software includes compilers, loaders, linkers and debuggers.
- It interacts with hardware

Application Software.

- Designed for an end users such as word processors, database system and spreadsheet programmers.
- Users can interacts with this
- MS office, outlook, notepad.

Major Goals of OS. ***

. → Execute users programs.

. → Make the computers system easy to use

. → Use the computers hardware in an efficient manners.

→ Manage and allocate all resources

→ Controls the execution of users programs and operations of I/O devices

Kernel *** : Central module of operating System.

. • Central module of OS

. • As small as possible

. • Part of OS that load first

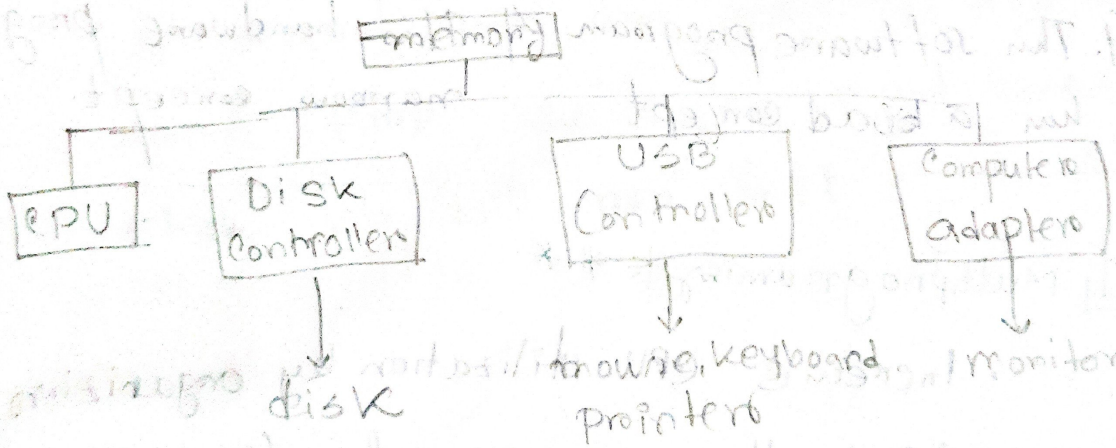
. • Provides all the services required by other parts

. Kernel code loaded protected area of memory.

Bootstrap Program ***

- When the computer starts, it is executed first
- Store in ROM or EEPROM, known as firmware
- It knows how to load OS and how to start executing
- Initializes all aspects of the system,
  from CPU register to device controller to memory contents

Computer System Organization



- Each devices has a local buffer which store data while it is being processed or transfered.
- CPU moves data from/to main memory to/from local memory

Device Drivers.	Device Controller
1. This is a software program.	1. This is a hardware program.
2. It helps to communicate with OS of different computers devices.	2. It helps to link between ingoing and ongoing signals of OS of computers.
3. It helps to interacting with different kinds of operating System.	3. It working as bridge between hardware and software
4. This software program has a broad concept	4. This hardware program has a narrow concept.

Multiprograming***

- Increase CPU utilization by organizing jobs.
- OS handle many job at a time. As main memory small it keeps job on the disk which there to be allocated in the main memory
- OS picks and begins to execute one of the jobs in memory.

Requirements of Multiprogramming.

→ Job scheduling.

- If a lot of jobs ready to run, but the system must choose among of them because there is no enough room for all of them.
- When OS select a job in job pool its loads that jobs into memory for execution.

→ CPU scheduling

- If several jobs are ready to run at the same time, the system must choose which job will run first.

System Call

- System calls provide an interface to the service made available by an operating system.
- System call is a mechanism that provides the interface between a process and the operating system.

Types of System Call

- Process Call
- File Manipulation
- Device Manipulation
- Information Manipulation
- Communication
- Protection.

System Programs.

- System programs also known as system utilities which provide a convenient environment for program development and execution.

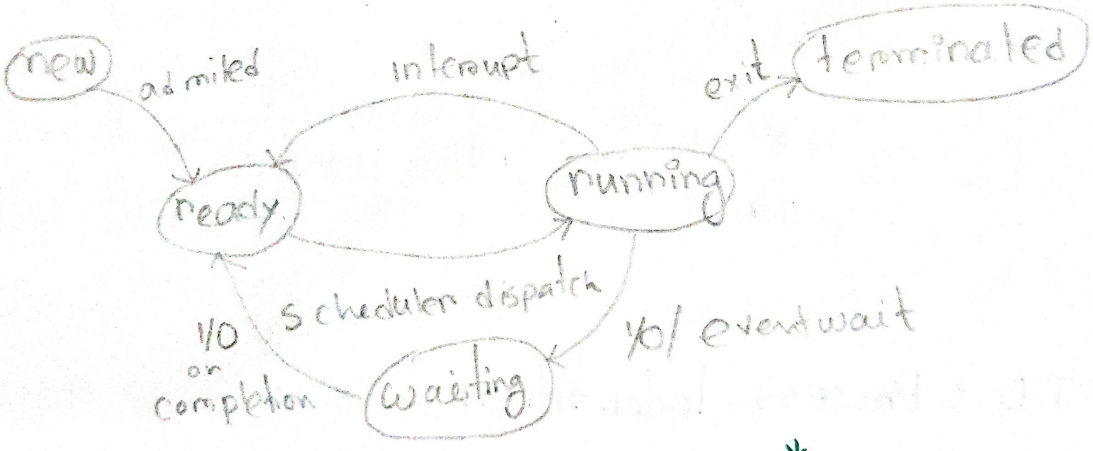
It provides.

→ file management
→ Status Information
→ file modification
→ Communication.

Difference Between program and process ***

Program	Process.
1. Program is set of Instruction	1. When a program is executed it is known as process
2. It is a passive entity	2. It is a active entity
3. It has longer lifespan	3. It has Limited lifespan.

Process State Diagram. ***

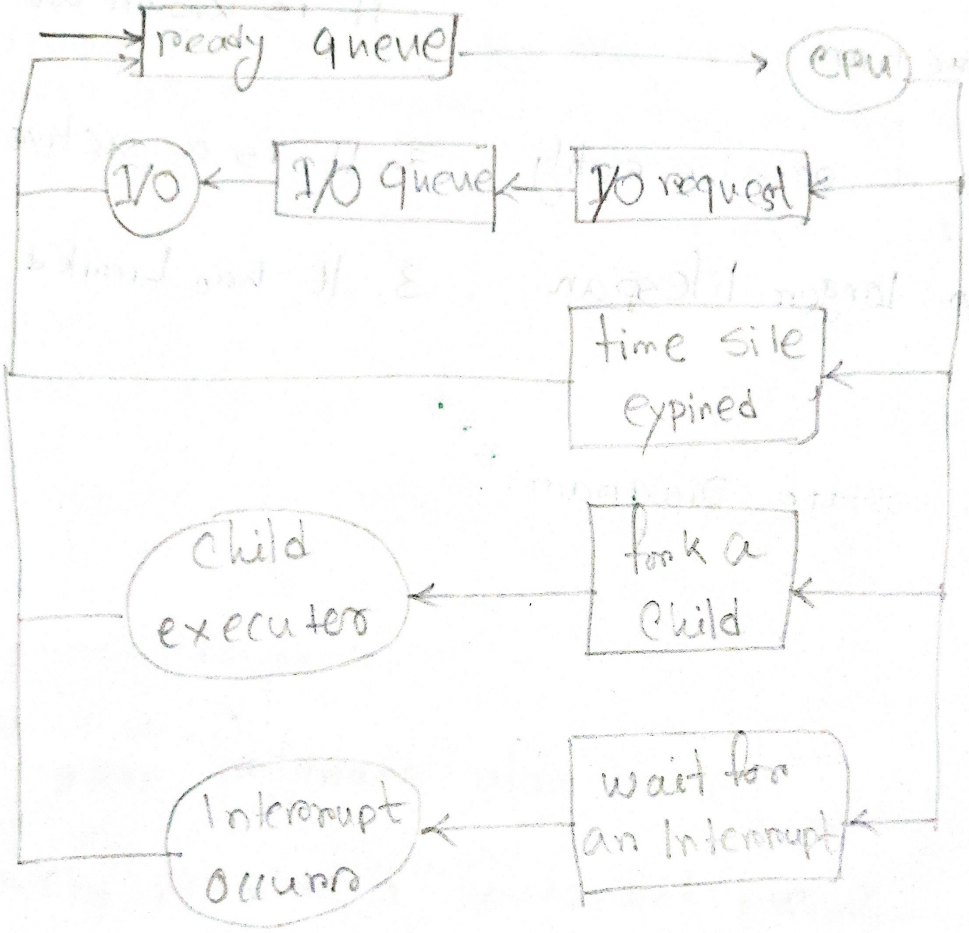


PCB : Process Control Block ***

PCB is a Data Structure to store information of process.

Attributes : Process Id, Process State, Priority, Program counter.

Queueing Diagram

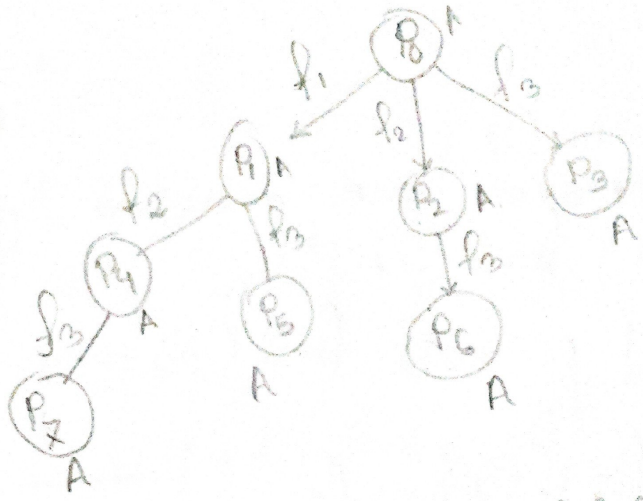


PID : Process Identifier

- A process is identified by a unique PID (process identifier) in the OS.

fork

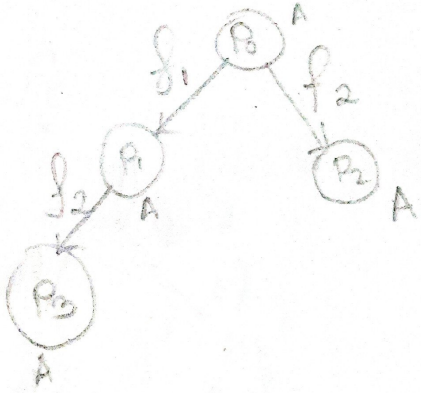
```
int main()
{
    fork(); → f1
    fork(); → f2
    fork(); → f3
    printf("A");
}
```



Output: AAAAAAAA

P0

```
int main()
{
    fork(); → f1
    fork(); → f2
    Printf("A");
}
```

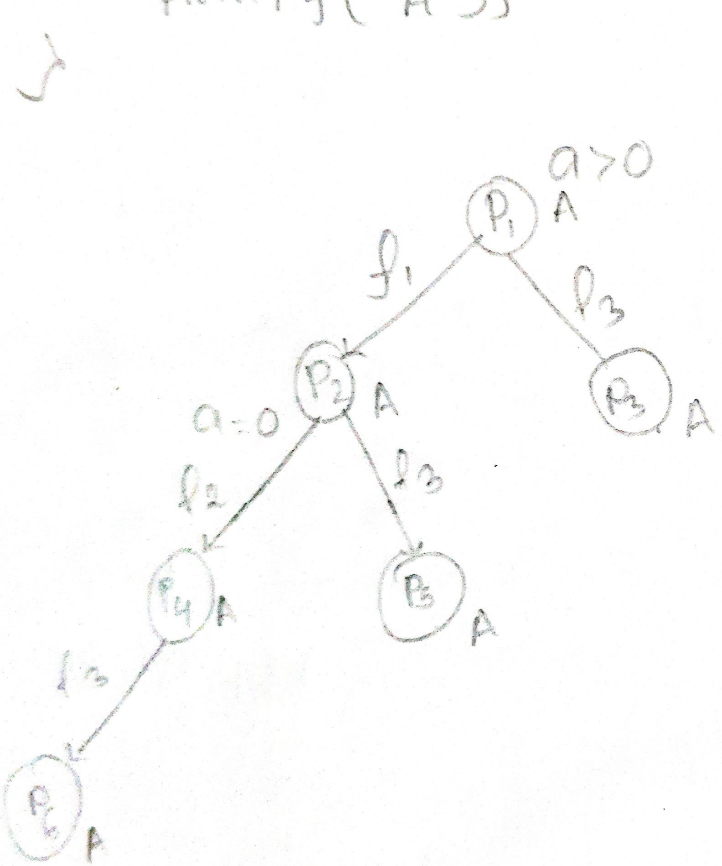


output: AAAA

P1

```
int main () {
    a = fork(); → f1
    if (a==0) fork(); → f2
    fork(); → f3
    printf("A");
}
```

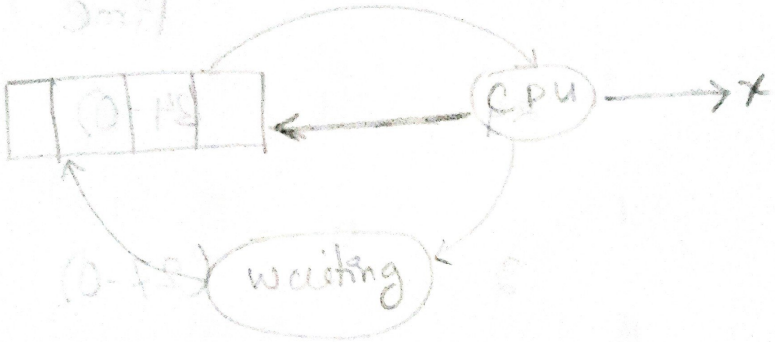
(*) Parrent এর
জন্য Parrent
হবে a<0
এবং Child এর
জন্য a=0 হবে



Output: "AAAAAA"

CPU Scheduler

- CPU scheduling decision may take place when.



Scheduling Criteria

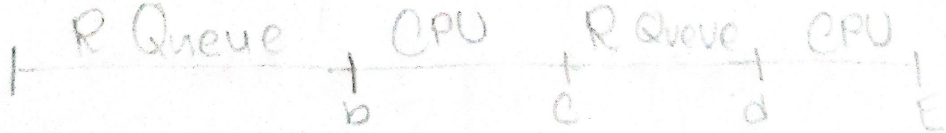
↑ → CPU Utilization

↑ → Throughput

↓ → Trun around Time: c-a

↓ → Waiting time : b-a

↓ → Response Time : b-a

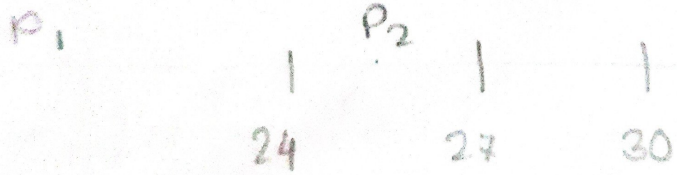


Trunaround Time : e-a
Waiting Time : (b-a) + (d-c)
Response Time: b-a

FCFS: First come first serve [Criteria: Arrival time]

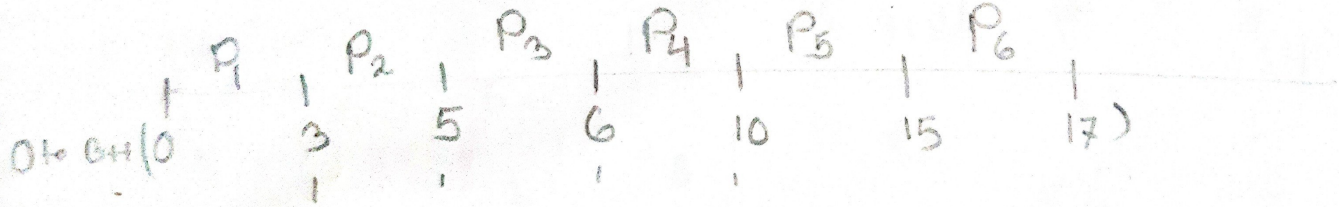
Process	Burst Time	Tournaround Time	Waiting Time
P_1	24	(24-0)	(0-0)
P_2	3	(27-0)	(24-0)
P_3	3	(30-0)	(27-0)

- Suppose that the process arrive in the order : P_1, P_2, P_3
- The gant chart for the schedule is.



Example

Process	A Time	B Time	T Time (End - ATime)	wTime Start - ATime
P_1	0	3	3	0
P_2	1	2	4	2
P_3	2	1	4	3
P_4	3	4	7	3
P_5	4	5	11	6
P_6	5	2	12	10

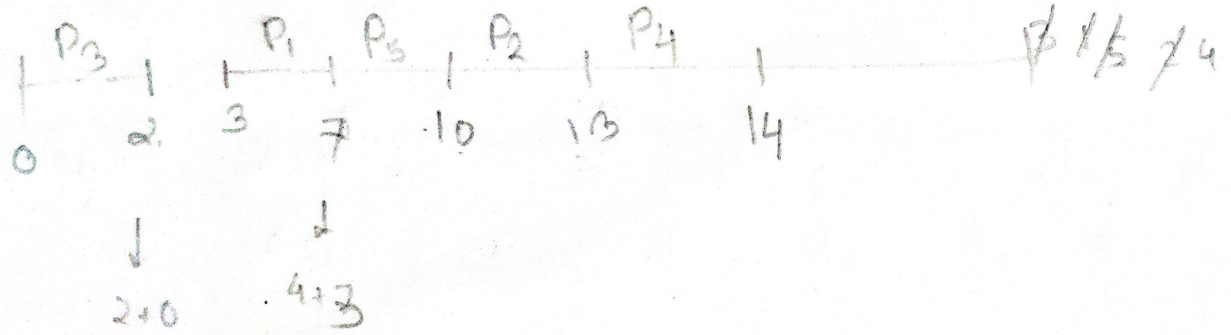


$$\text{Avg T. Time} = \frac{41}{6} = 6.8333$$

$$\text{Avg w. Time} = \frac{24}{6} = 8$$

Example

Process ID	A Time	B Time	T Time	W Time
P_1	3	4	4	0
P_2	5	3	8	5
P_3	0	2	2	0
P_4	5	1	9	8
P_5	4	3	6	3

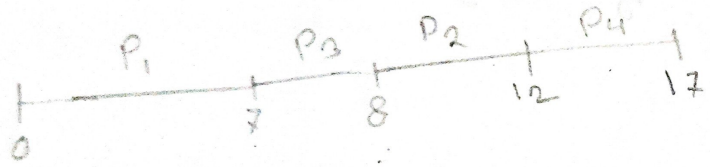


$$\text{Avg T Time} = \frac{29}{5} = 5.8$$

$$\text{Avg W Time} = \frac{16}{5} = 3.2.$$

Shortest Job First [Criteria: Burst time]

Process	A Time	B Time	Compne... Time	TAT	WT
P_1	0	7	7	7	0
P_2	2	4	12	10	6
P_3	4	1	8	4	3
P_4	5	5	17	12	7



$TAT = CT - AT$

$WT = TAT - BT$

Avg TAT = 8.25 ; Avg = 4

- Process

	A_T	B_T	C_T	T_T	W_T
P_1	2	6	9	7	1
P_2	5	2	11	6	4
P_3	1	8	23	22	14
P_4	0	3	3	3	0
P_5	4	4	15	11	7

Avg T_T = 9.8

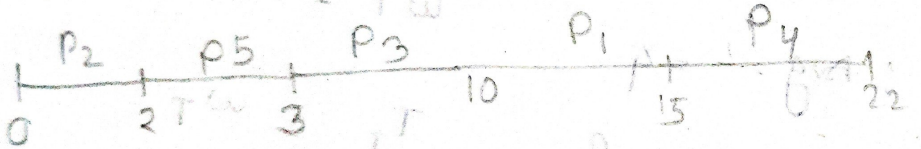
Avg W_T = 5.2

Process	A_T	B_T	C_T	T_T	W_T
P_1	4	5	15	11	6
P_2	0	2	2	2	0
P_3	1	5	10	9	4
P_4	6	7	22	16	9
P_5	2	3	5	3	0

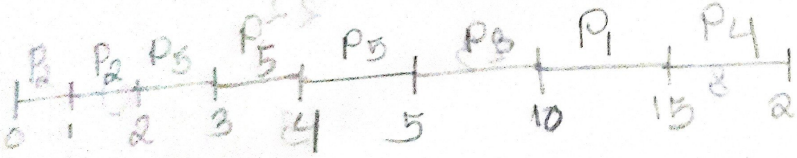
Avg $T_T = 8.2$

Avg $W_T = 3.8$

Non-Preemptive

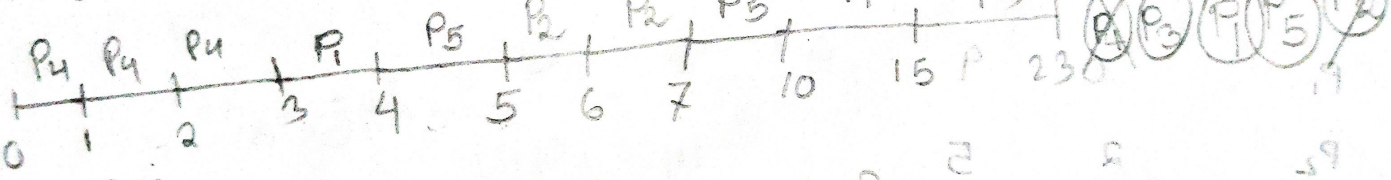


Preemptive

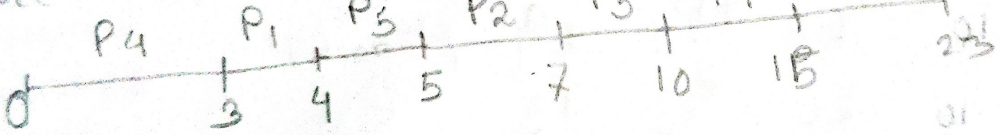


Process	A T	B T	C T	T T	W T
P_1	2	6 5			
P_2	5	2 1 0			
P_3	1	8			
P_4	0	3 2 1 0			
P_5	4	4 3			

Preemptive



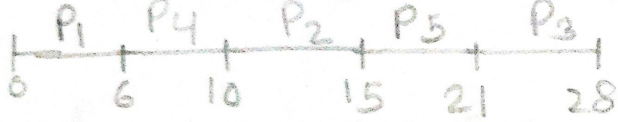
Non-Preemptive



Process	Arrival	B_T	C_T	T_{AT}	W_T
P_1	0	6	6	6	0
P_2	4	5	15	11	6
P_3	5	7	28	23	16
P_4	2	4	10	8	4
P_5	3	6	21	18	12

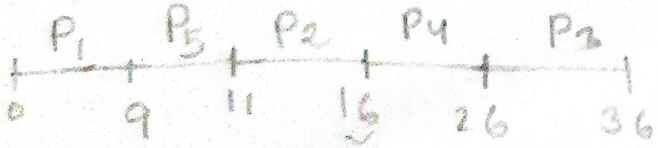
Avg T_{AT} = 13.2

Avg W_T = 7.6



Process	A_T	B_T	C_T	T_{AT}	W_T
P_1	0	9	9	9	0
P_2	2	5	16	14	9
P_3	2	10	36	34	24
P_4	1	10	26	25	15
P_5	3	2	11	8	6

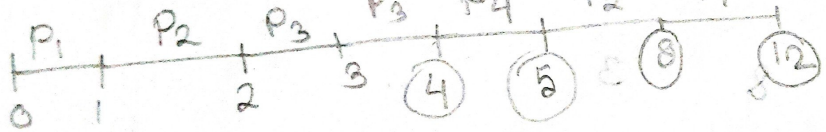
Avg T_{AT} =



Pre-emptive Priority Scheduling

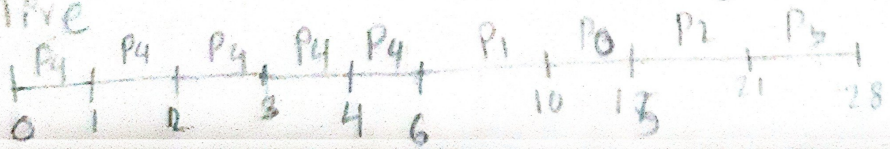
Process	Priority	A_T	B_T	C_T	T_T	W_T
P_1	10	0	~~5~~ 4	12	12	~~7~~
P_2	20	1	~~4~~ 3	8	~~7~~	3
✓ P_3	30	2	~~2~~ 1	4	2	0
✓ P_4	40	4	~~1~~ 0	5	1	0

Pre-emptive



Process	Priority	A_T	B_T	C_T	T_T	W_T
✓ P_0	2	3	5			
✓ P_1	1	2	4			
P_2	3	5	6			
P_3	4	1	7			
✓ P_4	3	0	~~5~~ 4			

Pre-emptive



Process	A_T	B_T	P_y	C_T	T_T	W_T
P_1	0	21	4			
P_2	8	18	5			
P_3	5	10	1			
P_4	8	13	3			
P_5	20	13	2			

Pre-emptive

Process	A_T	B_T	P_y	C_T	T_T	W_T
P_1	3	8	3	28	25	17
P_2	5	6	1	15	10	4
P_3	18	3	4	21	3	0
P_4	20	3	1	24	4	1
P_5	4	5	2	9	5	0