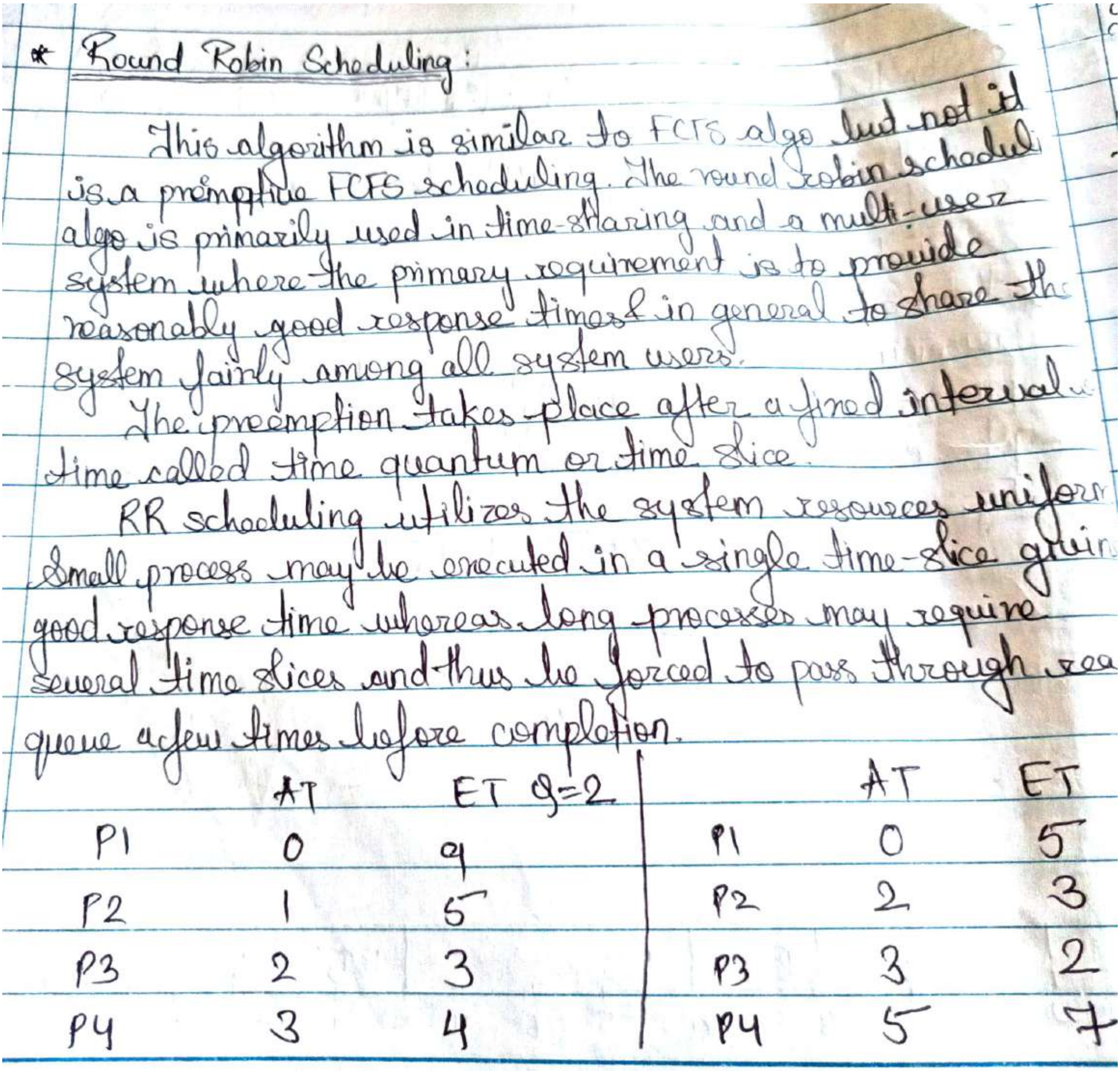


* Round Robin Scheduling :

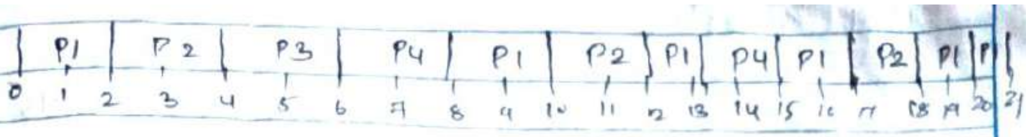
This algorithm is similar to FCTS algo but not it is a premptive FCFS scheduling. The round robin schedul algo is primarily used in time-sharing and a multi-user system where the primary requirement is to provide reasonably good response times & in general to share th system fairly among all system users.

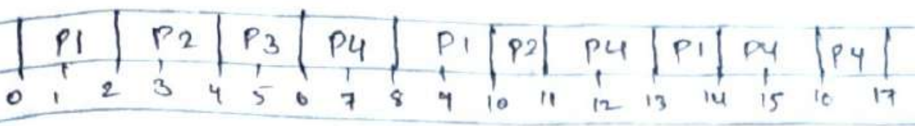
The preemption takes place after a fixed interval time called time quantum or time slice.

RR scheduling utilizes the system resources unifor Small process may be executed in a single time-slice givin good response time whereas long processes may require several time slices and thus be forced to pass through rea queue a few times before completion.

| | AT | ET q=2 |
|---|---|---|
| P1 | 0 | 9 |
| P2 | 1 | 5 |
| P3 | 2 | 3 |
| P4 | 3 | 4 |

| | AT | ET |
|---|---|---|
| P1 | 0 | 5 |
| P2 | 2 | 3 |
| P3 | 3 | 2 |
| P4 | 5 | 7 |





Rule1: 80% of the CPU bursts should be smaller than the time quantum.

Rule2: Content suitch time is nearly 10% of time quantum.

* Priority Scheduling:

In this algorithm, each process is given priority. The scheduler always picks up the highest priority process for its execution from the ready queue. Priorities can be defined either internally or externally.

Internal priorities:

The priority of a process is defined using some measurable internal factors like time limits, memory requirements, no. of open files.

External priorities:

They are set up by criteria external to the OS such as importance of the process, type & amount of funds paid.

→ Priority scheduling are of two types:

1) Pre-emptive priority based scheduling:

A newly arrived process's priority is compared with an already existing priority; if it is higher, the current process is pre-empted else process continues.

2) Non-preemptive priority based scheduling:

The newly arrived process is put at the head of the priority queue.

a) eg: Preemptive

| Process | Burst time | Priority | Arrival time | WT | TAT |
|---|---|---|---|---|---|
| P1 | 10 | 3 | 0 | 7 | [illegible] |
| P2 | 5 | 2 | 1 | 8 | 2 |
| P3 | 2 | 1 | 2 | 0 | |

where 1 - highest priority

Gantt chart:-

| P1 | P2 | P3 | P2 | P1 |
|---|---|---|---|---|
| 0 | 1 | 2 | 4 | 8 | 17 |

b) eg: Non-preemptive

| Process | Burst Time | Priority | Arrival time | TAT | W |
|---|---|---|---|---|---|
| P1 | 10 | 5 | 0 | 10 | C |
| P2 | 6 | 4 | 1 | 21 | 15 |
| P3 | 2 | 2 | 3 | 13 | 1 |
| P4 | 4 | 0 | 5 | 09 | 05 |

| P1 | P4 | P3 | P2 |
|---|---|---|---|
| 0 | 10 | 14 | 16 | 22 |

→ Drawback:

- Indefinitely blocking or Starvation:-
A process that is already to run but waiting for the CPU is considered to be blocked or starved in this algo since a low priority process ha to wait indefinitely for the CPU by a high priority process
A soln to this problem is aging. Aging is a techniqu of gradually increasing the priority of processes that wai in the system for a long time. Thus, the older processes attain high priority and are ensured of completion in a finite time.