Implementation of UNIX Dynamic Priority CPU Scheduling Algorithm

Objectives:

1. To understand how CPU scheduling works in a time-sharing system like UNIX.
2. To learn how process priorities are updated dynamically based on CPU usage.
3. To implement and simulate dynamic preemptive scheduling in C.

Theory Background:

UNIX uses a time-sharing scheduling algorithm where:

- Each process has a base (static) priority determined by its “nice” value.
- A dynamic priority is calculated as:

  Priority Priority usageDynamic Priority=Base Priority+CPU usage/2
- The process with lowest dynamic priority (i.e., highest importance) runs first.
- After every time slice (quantum), the priority is recalculated.

Problem Statement:

Write a C program to simulate UNIX dynamic priority CPU scheduling using the following rules:

1. Each process has:
    - Process ID
    - CPU burst time
    - Base priority (nice value)
    - CPU usage counter
2. At each time slice:
    - The scheduler selects the process with lowest dynamic priority.
    - That process runs for one time quantum.
    - Update its CPU usage and remaining burst time.
3. Continue until all processes complete.

Reference Code : dy_pr.c

Tasks to Perform (Within 1 Hour)

Part A — Implementation (10–15 min)

1. Compile and run the reference code.

2. Enter at least 3–5 processes with different burst times and priorities.
3. Observe the scheduling order and output.

Part B — Enhancement (10–15 min)

Modify the program to:

1. Calculate and print Waiting Time and Turnaround Time for each process.
2. Write a Gantt Chart showing CPU execution order (Follow the below example) .

<-------P0-----><-------p2------><-------p1------>
0 2 4 6

Part C — Enhancement (30–40 min)

Modify select_process() to select processes in round robin with SJF.
Use the time_quantum 3 time unit already in the program. Perform the tasks described in Part A and Part B.

Submission Format in Website:

Part A : The content of the terminal in text format for the reference code.

Part B : Waiting Time and Turnaround Time for each process for the reference code.

Part C :

1) Modified version of reference Code and the content of the terminal in text format.
2) Waiting Time and Turnaround Time for each process for modified Code.

Course Outcome : CO3