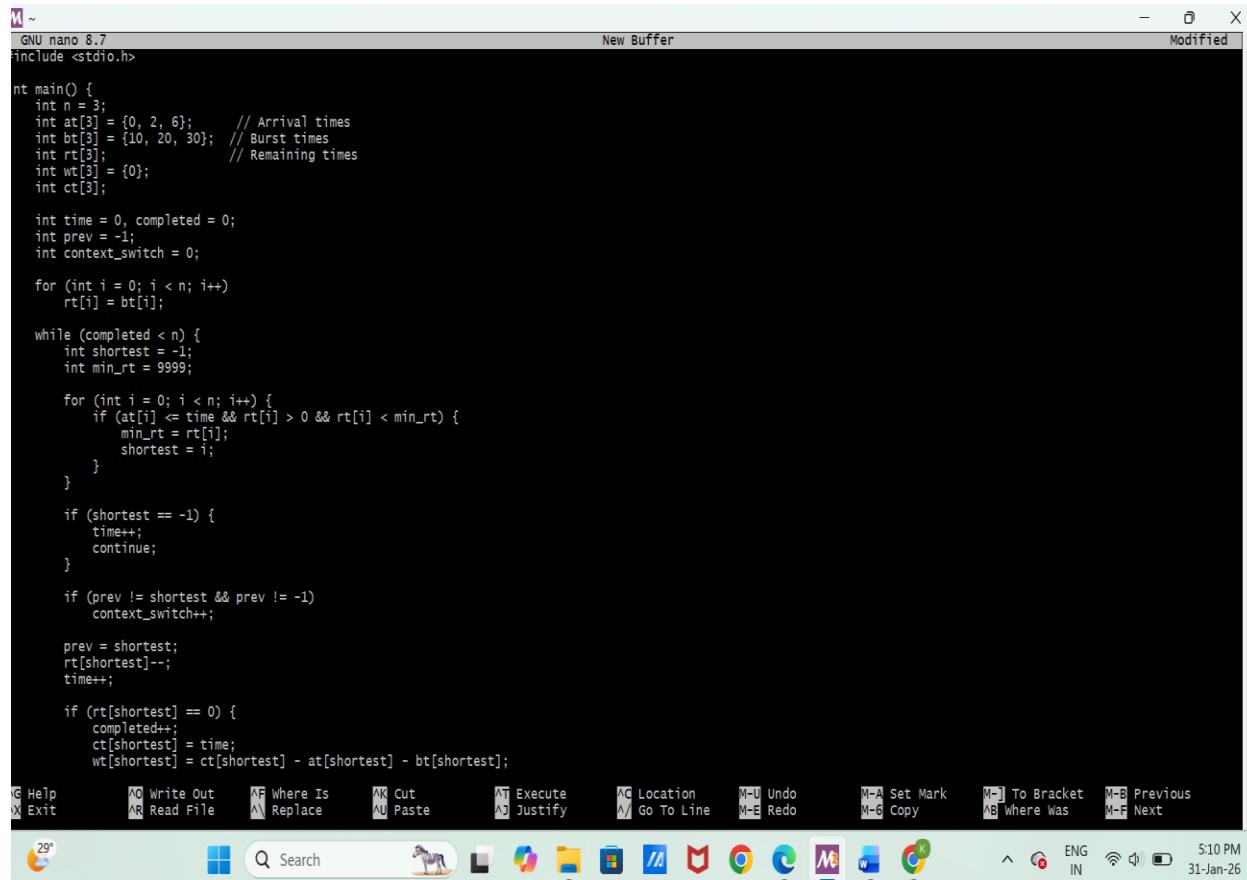


## 5.[Process management Based practical]

In an operating system three CPU-intensive processes are ready for execution, which require 10ns, 20ns and 30ns and arrival at times 0ns, 2ns and 6ns, respectively. Write a Program to calculate the total number of context switches needed if the operating system implements a shortest job first (preemptive) scheduling algorithm. Also calculate the average time for which the processes have to wait before getting the CPU.

### CODE:



The screenshot shows a terminal window titled "GNU nano 8.7" with the file name "Modified". The code implements a Shortest Job First (SJF) scheduling algorithm for three processes. It initializes arrays for arrival times (at), burst times (bt), remaining times (rt), waiting times (wt), and context switches (ct). It then enters a loop where it repeatedly finds the process with the shortest remaining time and executes it until its burst time is zero. It increments the completion time and the count of context switches. The process continues until all three processes have completed.

```
~  
GNU nano 8.7  
include <stdio.h>  
  
int main() {  
    int n = 3;  
    int at[3] = {0, 2, 6}; // Arrival times  
    int bt[3] = {10, 20, 30}; // Burst times  
    int rt[3]; // Remaining times  
    int wt[3] = {0};  
    int ct[3];  
  
    int time = 0, completed = 0;  
    int prev = -1;  
    int context_switch = 0;  
  
    for (int i = 0; i < n; i++)  
        rt[i] = bt[i];  
  
    while (completed < n) {  
        int shortest = -1;  
        int min_rt = 9999;  
  
        for (int i = 0; i < n; i++) {  
            if (at[i] <= time && rt[i] > 0 && rt[i] < min_rt) {  
                min_rt = rt[i];  
                shortest = i;  
            }  
        }  
  
        if (shortest == -1) {  
            time++;  
            continue;  
        }  
  
        if (prev != shortest && prev != -1)  
            context_switch++;  
  
        prev = shortest;  
        rt[shortest]--;  
        time++;  
  
        if (rt[shortest] == 0) {  
            completed++;  
            ct[shortest] = time;  
            wt[shortest] = ct[shortest] - at[shortest] - bt[shortest];  
        }  
    }  
}  
  
^G Help ^O Write Out ^F Where Is ^K Cut ^T Execute ^C Location ^U Undo  
^X Exit ^R Read File ^L Replace ^P Paste ^J Justify ^G Go To Line ^E Redo  
M-A Set Mark ^N-] To Bracket M-6 Copy ^B Where Was M-B Previous  
M-F Next
```

M ~

GNU nano 8.7

New Buffer

Modified

```
while (completed < n) {
    int shortest = -1;
    int min_rt = 9999;

    for (int i = 0; i < n; i++) {
        if (at[i] <= time && rt[i] > 0 && rt[i] < min_rt) {
            min_rt = rt[i];
            shortest = i;
        }
    }

    if (shortest == -1) {
        time++;
        continue;
    }

    if (prev != shortest && prev != -1)
        context_switch++;

    prev = shortest;
    rt[shortest]--;
    time++;

    if (rt[shortest] == 0) {
        completed++;
        ct[shortest] = time;
        wt[shortest] = ct[shortest] - at[shortest] - bt[shortest];
    }
}

float avg_wt = 0;
printf("\nProcess\tAT\tBT\tWT\n");
for (int i = 0; i < n; i++) {
    avg_wt += wt[i];
    printf("%d\t%d\t%d\t%d\n", i + 1, at[i], bt[i], wt[i]);
}

printf("\nTotal Context Switches = %d", context_switch);
printf("\nAverage Waiting Time = %.2f ns\n", avg_wt / n);

return 0;
}
```

Help    Write Out    Where Is    Cut    Execute    Location    Undo    Set Mark    To Bracket    Previous

Exit    Read File    Replace    Paste    Justify    Go To Line    Redo    Copy    where Was    Next

29

Search

EN IN

5:10 PM  
31-Jan-26

## Output:

```
M ~

user@SRUSHTI MSYS ~
$ nano sjf_preemptive.c

user@SRUSHTI MSYS ~
$ gcc sjf_preemptive.c -o sjf

user@SRUSHTI MSYS ~
$ ./sjf

Process AT      BT      WT
P1      0      10      0
P2      2      20      8
P3      6      30     24

Total Context Switches = 2
Average Waiting Time = 10.67 ns

user@SRUSHTI MSYS ~
$
```