

```
Documents  rms_with_preemption...  rms_with_preemption.c - /home/pi/Dc

View Document Project Build Tools Help

edf_with_preemption.c x rms_with_preemption.c x pthread.c x

1  #include <stdio.h>
2  #include <math.h>
3  #include <stdlib.h>
4
5  // The code below simulates how RMS algorithm works with processe A and B.
6  // Please read it in detail and understand how it works.
7  // Add your own code to make it be able to handle preemptive situations.
8  // Realize EDF algorithm based on this implementation.
9
10 int main() {
11     int periodA, periodB;           // period (relative deadline) of A and B
12     int execTimeA, execTimeB;       // worst-case execution time of A and B
13     int absDeadlineA, absDeadlineB; // absolute deadline of A and B
14     float cpuUtil;                  // CPU utilization
15     int jA = -1, jB = -1;           // indices of jobs
16     int doA = 0, doB = 0;           // switches between A and B (e.g. doA == 1 && doB == 0, run A)
17     int tA = 0, tB = 0;             // accumulated execution time
18     int T;                           // simulated time
19     double cpuA;
20     double cpuB;
21     double Ub;
22
23     // Input handling and reading
24     printf("\t\t\t-----\n");
25     printf("\t\t\tRate Monotonic Schedule (RMS) Algorithm\n");
26     printf("\t\t\t-----\n");
27     printf("please input period and execution for A process\ndefault: 25, 10: ");
28     scanf("%d", &periodA, &execTimeA);
29     printf("please input period and execution for B process\ndefault: 60, 15: ");
30     scanf("%d", &periodB, &execTimeB);
31
32     // Your code here to calculate & print CPU utilization
33     // Hint: use the definition of utilization
34     cpuA = (float)execTimeA/periodA;
35     cpuB = (float)execTimeB/periodB;
36     cpuUtil = cpuA + cpuB;
37     // End of your code
38     printf("CPU Utilization : %.6f\n", cpuUtil);
39
40     absDeadlineA = periodA, absDeadlineB = periodB;
41     printf("\nsimulation started\n");
42
43     for (T = 0; T <= 200; T++) {
44         // Your code here to check if CPU can schedule the task set
45         if (cpuUtil > 1) { printf("CPU cannot schedule task set\n");
46             return EXIT_FAILURE;
47         }
48         // End of your code
49
50         // process A is done
51         if (tA == execTimeA && doA == 1) {
52             printf("when T=%d, ", T);
53             printf("process A%d is done\n", jA);
54             doA = 0;
55             // resume possibly suspended process B
56             if (tB < execTimeB) {
57                 printf("when T=%d, ", T);
58                 printf("program switched to run process B%d\n", jB);
59                 doB = 1;
60             }
61         }
62     }
63 }
```

58 col: 115 sel: 0 INS SP mode: LF encoding: UTF-8 filetype: C scope: main

ents rms_with_preemption...

rms_with_preemption.c - /home/pi/Documents - Geany

ant Project Build Tools Help



emption.c x rms_with_preemption.c x pthread.c x

```
printf("when T=%d, ", T);
printf("program switched to run process B!\n", jB);
doB = 1;
}
}

// process B is done
if (tB == execTimeB && doB == 1) {
    printf("when T=%d, ", T);
    printf("process B!\n", jB);
    doB = 0;
    // resume possibly suspended process A
    if (tA < execTimeA) {
        printf("when T=%d, ", T);
        printf("program switched to run process A!\n", jA);
        doA = 1;
    }
}

// new instances of process A and B are generated together
if (T % periodA == 0 && T % periodB == 0) {
    printf("when T=%d, process A! and B! are generated together\n", T, ++jA, ++jB);
    absDeadlineA = T + periodA;
    absDeadlineB = T + periodB;
    // RMS: higher rate (1/period) ==> higher priority
    if (periodA <= periodB) {
        printf("when T=%d, program switched to run process A!\n", T, jA);
        doA = 1;
        doB = 0;
    } else {
        printf("when T=%d, program switched to run process B!\n", T, jB);
        doA = 0;
        doB = 1;
    }
    tA = 0;
    tB = 0;
}

// a new instance of process A is generated
if (T % periodA == 0 && T % periodB != 0) {
    printf("when T=%d, process A! is generated\n", T, ++jA);
    absDeadlineA = T + periodA;
    tA = 0;
    if (tB < execTimeB) { // process B is unfinished yet
        // Your code here to handle preemption
        // Please print out the decision
        if (periodA <= periodB) {
            if (T + execTimeA + execTimeB - tB > absDeadlineB) { printf("Process B missed deadline! Not schedulable!\n"); return EXIT_FAILURE; }
            printf("when T=%d, program switched to run process A!\n", T, jA);
            doB = 0; doA = 1;
        }

        if (T > absDeadlineB) { printf("process B missed deadline! Not schedulable!\n"); return EXIT_FAILURE; }
    }

    // End of your code
} else { // process B is done, just run A
    printf("when T=%d, program switched to run process A!\n", T, jA);
```

sel: 0 INS SP mode: LF encoding: UTF-8 filetype: C scope: main

DELL


```
rms_with_preemption...
rms_with_preemption.c - /home/pi/Documents - Geany

Project Build Tools Help

nption.c: x rms_with_preemption.c: x pthread.c: x

tA = 0;
if (tB < execTimeB) { // process B is unfinished yet
    // Your code here to handle preemption
    // Please print out the decision
    if (periodA <= periodB) {

        if (T+execTimeA + execTimeB- tB > absDeadlineB) { printf("Process B missed deadline! Not schedulable!\n"); return EXIT_FAILURE; }
        printf("when T=%d, program switched to run process A%d!\n", T, jA);
        doB = 0; doA = 1;

    }

    if (T>absDeadlineB) { printf("process B missed deadline! Not schedulable!\n"); return EXIT_FAILURE; }

    // End of your code
} else { // process B is done, just run A
    printf("when T=%d, program switched to run process A%d!\n", T, jA);
    doA = 1;
}

// a new instance of process B is generated
if (T % periodA != 0 && T % periodB == 0) {
    printf("when T=%d, process B%d is generated!\n", T, ++jB);
    absDeadlineB = T + periodB;
    tB = 0;
    if (tA < execTimeA) { // process A is unfinished yet
        // Your code here to handle preemption
        // Please print out the decision
        if (periodB < periodA) {

            if (T+execTimeB + execTimeA- tA > absDeadlineA) { printf("Process A missed deadline! Not schedulable!\n"); return EXIT_FAILURE; }
            printf("when T=%d, program switched to run process B%d!\n", T, jB);
            doB = 1; doA = 0;

        }

        if (T>absDeadlineA) { printf("Process A missed deadline! Not schedulable!\n"); return EXIT_FAILURE; }

        // End of your code
    } else { // process A is done, just run B
        printf("when T=%d, program switched to run process B%d!\n", T, jB);
        doB = 1;
    }
}

// accumulate running time of process A or B
if (doA) {
    tA++;
}
if (doB) {
    tB++;
}

}

sel: 0  INS  SP  mode: LF  encoding: UTF-8  filetype: C  , scope: unknown
```