

# CSE231 – Operating Systems

## Assignment-3

**Name:** Ananya Lohani  
**Roll Number:** 2019018  
**Branch:** CSE

### **Description:**

In this assignment, I have modified the `sched_entity` structure to contain a new field `rt_nice` which is the soft real-time requirement of a process (a value of `x` as `rt_nice` means that the process must receive at least `x` units of time-slice). To give priority to processes with soft real-time requirements I have modified the CFS scheduler to schedule tasks on the basis of `rt_nice` first, and `vruntime` second.

### **rt\_nice System Call implementation:**

I added a new field `rt_nice` to the `sched_entity` struct in `/kernel/sched/sched.h` and initialised it to 0 in the `__sched_fork()` function in `/kernel/sched/core.c`. I have added a system call `rt_nice` which takes two arguments: **PID of the process** and the **soft real-time requirement**. The system call gets the task corresponding to the given PID and sets the `rt_nice` field of its `sched_entity` to the given value. The system call number of `sys_rt_nice` is 441.

### **Modified CFS implementation:**

The CFS scheduler of Linux has been modified to give priority to the `rt_nice` values before giving priority to the `vruntime` of processes in the run-queue.

### **Functions modified:-**

- **entity\_before():** This function acts as a comparator between the `vruntime` of 2 `sched_entity` struct pointers. I added the code that compares the `rt_nice` values of the pointers first, and if both pointers' `rt_nice` is equal to 0, it compares their `vruntime`.
- **\_\_pick\_next\_entity():** This function picks the next process to be executed. I modified it to select the process in the run-queue with the **maximum** `rt_nice` to be executed. If no process has `rt_nice`

greater than zero, it will execute the next process according to `vruntime`.

- **update\_curr()**: If the `rt_nice` value of a `sched_entity` is greater than 0, it updates `rt_nice` by subtracting from it the amount of time that the process ran.

## Errors handled:-

Errors returned by the system call are handled by `perror()` in `test` files.

- **EINVAL**: If an invalid pid was given(pid not between 1 and 2147483647) or if invalid value for `rt_nice` was given(<0).
- **ESRCH**: If there's no process corresponding to the given pid.

## Testing the Scheduler:

The working of the scheduler can be tested by running `test-cases/test.c`. The program forks 5 processes, executes a loop of 2000000000 and displays the time taken in two cases: when there are no soft real-time requirements(`rt_nice = 0`) and when there are non-zero soft real-time requirements(`rt_nice > 0`). The file `test-cases/rt_nice.c` included in `test.c` calls the system call `rt_nice` and supplies the pid and soft real-time requirements as the argument.

## Expected output:

The processes with `rt_nice > 0` take **less time** than processes with `rt_nice = 0`. The expected output is as follows:

```
./test
Time taken with rt_nice = 0:
Process 3, PID: 1253, Time: 6.878472s
Process 4, PID: 1254, Time: 6.963350s
Process 1, PID: 1251, Time: 6.983581s
Process 5, PID: 1255, Time: 7.037397s
Process 2, PID: 1252, Time: 7.200182s
Time taken with rt_nice > 0:
Process 1, PID: 1266, rt_nice: 10, Time: 5.639804s
Process 5, PID: 1270, rt_nice: 50, Time: 5.647697s
Process 3, PID: 1268, rt_nice: 30, Time: 5.650595s
Process 2, PID: 1267, rt_nice: 20, Time: 5.651237s
Process 4, PID: 1269, rt_nice: 40, Time: 5.588995s
```

**Note:** This modified CFS would only work 80-90% of the time because there may be page faults and cache misses that cannot be controlled by the scheduler.