# Operating Systems
# Assignment 2 – *Easy*

**Instructions:**

1. The assignment has to be done in a group of 2 members.
2. You can use Piazza for any queries related to the assignment; avoid asking queries on the last day.

## 1 Real Time Scheduling

In real-time systems, tasks must be completed within strict time limits, known as deadlines. If meeting a deadline is crucial to the running of the system and failure to do so may have catastrophic effects, that deadline is considered *hard*. If satisfying time constraints is desirable, yet missing a deadline would not result in significant consequences, then that deadline is termed *soft*.

To enable the execution of real-time jobs in xv6, you need to add the following real-time scheduling policies: the earliest deadline first (EDF) and rate monotonic (RM) scheduling algorithms. The current scheduler in xv6 is a round robin scheduler. The scheduler code for xv6 may be found in **proc.c** and its accompanying header file, **proc.h**.

### 1.1 System Specification:

In this assignment, you need to use xv6 with the following specifications:

1. The system shall use a single core. To enable xv6 to run on a single core, you need to **set** the **CPUS** variable to **1** in the Makefile.
2. Assume a set of $n$ pre-emptable tasks $\mathcal{T}$, such that the tasks do not share resources, and no precedence ordering exists among the tasks.
3. By default, all processes are initialized with the default xv6 scheduling policy. To change the scheduling policy of a process to the real-time scheduling policy (EDF or RM), you need to implement the *sys_sched_policy* system call. The signature of the system call is as follows:

   ```
   int sys_sched_policy(int pid, int policy)
   ```

   - *pid:* pid of the process.
   - *policy:* If 0 is given, the scheduling policy is set to EDF. If 1 is given, set the scheduling policy to RM.

If the augmented (new) set of processes is schedulable, it should return 0; otherwise, it should return -22. You can read about schedulability checks from the following links: EDF and RMA.

***Note: To break ties between processes with the same static priority, you can choose the process with the smaller pid.***

4. The xv6 kernel may safely terminate a process when it fully executes (executes for a pre-specified time). With the *sys_exec_time* system call, the programmer will be able to define the duration of a process's execution. The signature of the system call is as follows:

   ```
   int sys_exec_time(int pid, int exec_time)
   ```

   - *pid:* pid of the process.
   - *exec_time:* number of ticks for which the process should run.

   If the system call was successful, it should return 0; otherwise, it should return -22.

5. To specify the relative deadline of the task (after it has been released) in the EDF scheduling policy, you must implement the *sys_deadline* system call. The signature of the system call is as follows:

   ```
   int sys_deadline(int pid, int deadline)
   ```

   - *pid:* pid of the process.
   - *deadline:* deadline of the process (in ticks) relative to its arrival time.

   If the system call was successful, it should return 0; otherwise, it should return -22.

6. To specify the rate (reciprocal of the period) for a process in the RM scheduling policy, you must implement the *sys_rate* system call. The signature of the system call is as follows:

   ```
   int sys_rate(int pid, int rate)
   ```

   - *pid:* pid of the process.
   - *rate:* rate of the process (instances per second).

   If the system call was successful, it should return 0; otherwise, it should return -22. You can assume that the value of the rate lies in the range $[1, 30]$ and the value of the weight (type: integer) lies in the range $[1, 3]$ (lower the weight, higher is the priority.). To derive the weight ($w$) for a process with rate ($r$) use the following equation:

$$w = max\left(1, \left\lceil \left(\frac{30 - r}{29}\right) * 3 \right\rceil\right) \tag{1}$$

## 2 Report: 10 Marks

**Page limit: 10**
The report should clearly mention the implementation methodology for all the real-time scheduling policies. Showing small, representative code snippets in the report is alright, additionally, the pseudocode should also suffice.

- Implementation of the EDF and RM scheduling policies.
- Any other details that are relevant to the implementation.

Submit a PDF file with the name *A2_report.pdf* that contains all relevant details. Also, you must **ensure** that the group members' **entry numbers** are listed on the **cover page**.

## 3 Submission Instructions

- We will run MOSS on the submissions. Any cheating will result in a zero in the assignment, a penalty as per the course policy and possibly much stricter penalties (including a fail grade).
- There will be NO demo for assignment 2. Your code will be evaluated using a check script (check.sh) on hidden test cases and marks will be awarded based on that. You can find the test scripts here.

How to submit:

1. Copy your report to the xv6 root directory.
2. Then, in the root directory run the following commands:

```
make clean
tar czvf \
  assignment2_easy_<entryNumber1>_<entryNumber2>.tar.gz *
```

   This will create a tarball with the name, $assignment2\_easy\_ < entryNumber1 > \_ < entryNumber2 > .tar.gz$ in the same directory that contains all the xv6 files and the report PDF document. Entry number format: 2020CSZ2445 (*All English letters will be in capitals in the entry number.*). **Only one** member of the group is required to **submit** this tarball on Moodle.

3. Please note that if the report is missing in the root directory, then no marks will be awarded for the report.
4. If you are attempting the assignment as an individual, you do not need to mention the entryNumber_2 field.

How to validate:

1. Find and replace the following lines in the trap.c file of xv6:

```
// Force process to give up CPU on clock tick.
// If interrupts were on while locks held, would need to
// check nlock.
if(myproc() && myproc()->state == RUNNING &&
    tf->trapno == T_IRQ0+IRQ_TIMER)
{
    if((myproc()->sched_policy >= 0) &&
       (myproc()->elapsed_time >= myproc()->exec_time))
    {
      cprintf("The completed process has pid: %d\n",
            myproc()->pid);
      exit();
    }
    else
      yield();
}
```

Note: You may need to change the terminate condition according to your implementation.

2. Run the following commands to validate your submission:

```
sudo apt install expect
mkdir check_scripts
tar xzvf check_scripts.tar.gz -C check_scripts
cp assignment2_easy_<entryNumber1>_<entryNumber2>.tar.gz \
check_scripts
cd check_scripts
bash check.sh \
assignment2_easy_<entryNumber1>_<entryNumber2>.tar.gz
```