تمرین دوم – سیستم های نهفته

کیوان ایپچی حق - ۹۸۳۱۰۷۳

پیاده سازی

پیاده سازی scheduler و سیستم RTOS از ابتدا انجام شده :

پیاده سازی Preemptive EDF

برای این الگوریتم تنها نزدیک ترین زمان deadline به صورت داینامیک اهمیت دارد و نوع تسک اهمیت چندانی ندارد (جز تسک از نوع interrupt که اولویت همیشه بالاترین را دارد و برای آن در کد استثنا قائل شدیم).

```python
def edf_preemptive(self, time: int):
    """ Preemptive Earliest Deadline First (EDF) scheduler """

    all_tasks = self.taskset.get_all_tasks()    # Get ready tasks
    all_tasks_by_deadlines = sorted(all_tasks, key=lambda x: x.next_deadline(time))  # Sort tasks by deadlines

    current_task = None
    for task in all_tasks_by_deadlines:
        task.preflight(time)

        # Interrupt (very hard deadline)
        if task.is_ready and current_task is None and task.is_type_interrupt:
            task.set_state(TaskState.RUNNING, save=True)
            current_task = task
            continue

        # EDF doesn't really care about the type of the task
        if (task.is_ready or task.is_running) and current_task is None:
            task.set_state(TaskState.RUNNING)
            current_task = task

        task.save_state()   # Save the state of the task

    return current_task
```

پیاده سازی Non-Preemptive EDF

این الگوریتم نیز مشابه الگوریتم بالا بوده با این تفاوت که تسک در حالت RUNNING، باید RUNNING باقی بماند تا اجرا آن تکمیل شود.

```python
def edf_non_preemptive(self, time: int):
    """ None Preemptive Earliest Deadline First (EDF) scheduler """

    all_tasks = self.taskset.get_all_tasks()    # Get ready tasks
    all_tasks_by_deadlines = sorted(all_tasks, key=lambda x: x.next_deadline(time))  # Sort tasks by deadlines

    # Continue current running task
    current_task = None
    for task in all_tasks_by_deadlines:
        task.preflight(time)
        if task.is_running:
            current_task = task
            break

    for task in all_tasks_by_deadlines:
        task.preflight(time)

        # Interrupt (very hard deadline)
        if task.is_ready and current_task is None and task.is_type_interrupt:
            task.set_state(TaskState.RUNNING, save=True)
            current_task = task
            continue

        # EDF doesn't really care about the type of the task
        if (task.is_ready or task.is_running) and current_task is None:
            task.set_state(TaskState.RUNNING)
            current_task = task

        task.save_state()   # Save the state of the task

    return current_task
```

تمرین دوم – سیستم های نهفته

کیوان ایپچی حق - ۹۸۳۱۰۷۳

پیاده سازی RM

این الگوریتم static بوده و برای تسک های hard deadline (جز interrupt که همیشه بالاترین اولویت است) اهمیت بالاتری قائل است.

این الگوریتم بر اساس کوتاه ترین Period کار می‌کند و از آن جایی که static بوده، بر خلاف EDF در حال اجرا زمان period ارزیابی نمیشود.

```python
def rm(self, time: int):
    """ Rate Monotonic (RM) scheduler """

    all_tasks = self.taskset.get_all_tasks()    # Get ready tasks
    all_tasks_by_periods = sorted(all_tasks, key=lambda x: x.period)  # Sort tasks by their periods

    current_task = None
    for task in all_tasks_by_periods:
        task.preflight(time)

        # Interrupt (very hard deadline)
        if task.is_ready and current_task is None and task.is_type_interrupt:
            task.set_state(TaskState.RUNNING, save=True)
            current_task = task
            continue

        # Sporadic (harder deadline)
        if task.is_ready and current_task is None and task.is_type_sporadic:
            task.set_state(TaskState.RUNNING, save=True)
            current_task = task
            continue

        # Periodic or aperiodic tasks (no clear indication which has higher priority!)
        if (task.is_ready or task.is_running) and current_task is None:
            task.set_state(TaskState.RUNNING)
            current_task = task

        task.save_state()   # Save the state of the task

    return current_task
```

پیاده سازی DM

این الگوریتم هم مانند RM اما بر روی نزدیک ترین ددلاین به صورت static کار می‌کند. الگوریتم ها تقریبا شبیه به هم هستند جز در پارامتر های تصمیم گیری آنها.

```python
def dm(self, time: int):
    """ Deadline Monotonic (DM) scheduler """

    all_tasks = self.taskset.get_all_tasks()    # Get ready tasks
    all_tasks_by_deadlines = sorted(all_tasks, key=lambda x: x.next_deadline(time))  # Sort tasks by deadlines

    current_task = None
    for task in all_tasks_by_deadlines:
        task.preflight(time)

        # Interrupt (very hard deadline)
        if task.is_ready and current_task is None and task.is_type_interrupt:
            task.set_state(TaskState.RUNNING, save=True)
            current_task = task
            continue

        # Sporadic (harder deadline)
        if task.is_ready and current_task is None and task.is_type_sporadic:
            task.set_state(TaskState.RUNNING, save=True)
            current_task = task
            continue

        # Sporadic, aperiodic or periodic (EDF doesn't really care about the type of the task)
        if (task.is_ready or task.is_running) and current_task is None:
            task.set_state(TaskState.RUNNING)
            current_task = task

        task.save_state()   # Save the state of the task

    return current_task
```

تمرین دوم – سیستم های نهفته

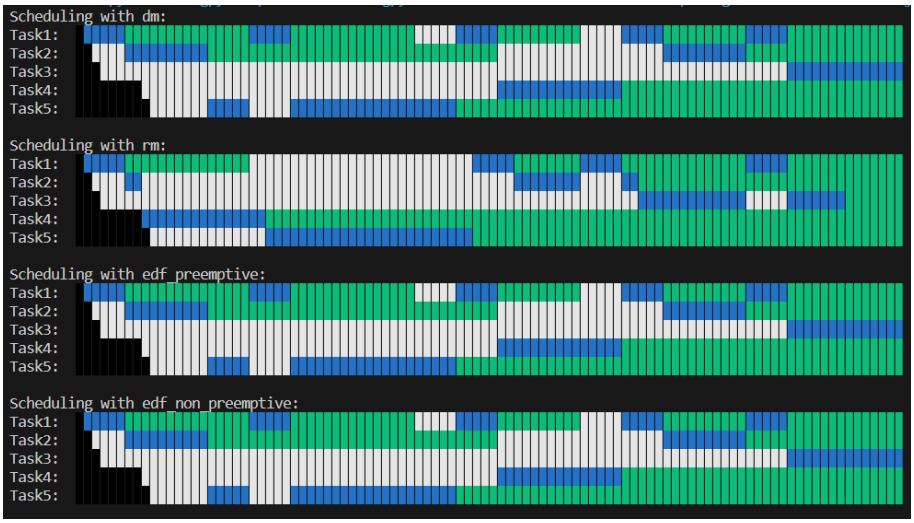
کیوان ایپچی حق - ۹۸۳۱۰۷۳

در هر تسک ست داده شده کدام الگوریتم مناسب تر است؟

بر حسب شبیه سازی های انجام شده توسط کد خروجی زیر حاصل شد:

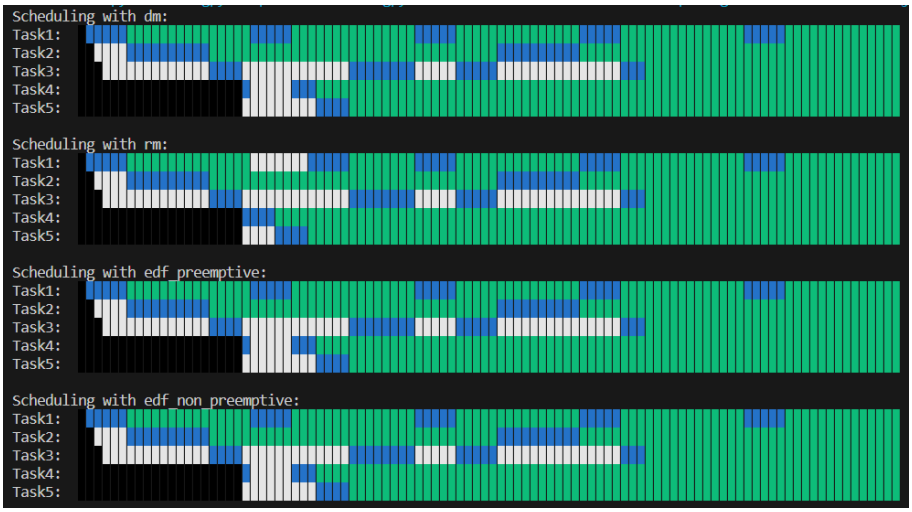
Task Set / Optimality	RM	DM	NP-EDF	P-EDF
Task1	Yes	Yes	Yes	Yes
Task2	Yes	Yes	Yes	Yes
Task interrupt	Yes	Yes	Yes	Yes

خروجی task1



همانطور که با SimSo تایید شد (خروجی یکسان است)، RM برای تسک اول فیل شده و DM و EDF موفق میشوند.

خروجی task2

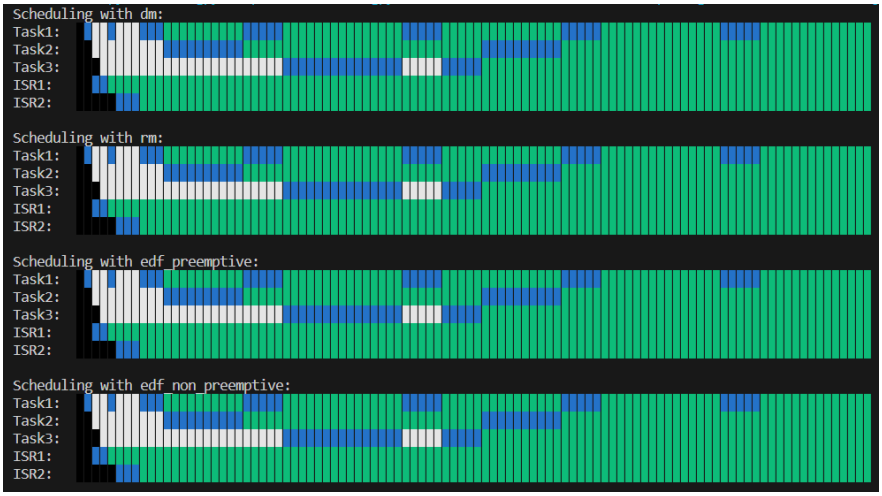


الگوریتم RM تعداد context switch کمتری دارد.

تمرین دوم – سیستم های نهفته

کیوان ایپچی حق - ۹۸۳۱۰۷۳

خروجی tasks_interrupts



برای این تسک ست، interrupt ها ابتدا انجام میشوند زیرا اولویت بالاترین را دارند. بین ۳ تسک باقی مانده فرقی نمیکند و همه میتوانند زمانبندی کنند.