# Assignment 4 (88 points)

In this assignment, you will modify a preemptive deadline monotonic (DM) schedule simulator to also include support for shared resources. I am providing a visualizer that now has support for depicting when resources are held. As this is just a simulation, you will need to specify when the resource-holding computations of each task occur. Note that this does not reflect how a real system would execute; when the resource-requiring computations occur would be based on the code executed for that job, and may even differ between jobs of the same task. For your simulation, however, assume that each job requires at most one resource and that all CPU-only computation is completed before the CPU+Resource execution.

## Part 1: Setting up your workspace.

In your home directory on the server, make a new directory called `HW4`. All of the files you intend to submit must be in this folder. Provided files are in `/home/shared/HW4/`.

Take a look at the files `visualizer.py`, `sample-input.json`, `sample-output.json`, and `sample.pdf` to see the format expected by the visualizer to show executions with resources.

Resource requirements should now be listed within the tasks file, as shown in the sample file. The key 'resource' should be used for the name of the resource, and the key 'cs_len' for the critical section length.

There is also a script posted that can help speed your testing. To run the script, type:

```
./testing.sh
```

If it complains that you do not have the correct permissions, change the permissions:

```
chmod +x testing.sh
```

You should only need to change the permissions once.

## Part 2: The assignment.

### A - Creating testing data (20 points)

For each of the following, create a task set of periodic tasks which showcases the differences in the following scenarios. Pick and include a `max_t` that shows at least one job completion for each task. Describe the differences that should be observed in `README.txt`.

1. `test1.json` - Create a sample task set with four tasks that shows at least three differences in execution based on the use of shared resources protected with the NPP. That is, the schedule generated by DM with the given periods and computation times should be different from that

generated by DM+NPP with the same periods and computation times plus the information about how long each resource is required.
2. `test2.json` - Create a different sample task set with four tasks that shows at least three differences in execution with the PIP instead of the NPP.

Grading:

- For each test:
  - (2 points) Readable and correct formatting of JSON. (That is, spacing within file allows easy reading of each task, and the file follows formatting of tasks expected by python scripts from the first two assignments.)
  - (4 points) A description in `README.txt` of how the schedules ought to be different. Point out at least three executions that differ between the protocols.
  - (4 points) The tasks themselves, that they follow the instructions (show a job completion for each task, etc.) and should cause the described differences.

## B - Non-Preemptive Protocol (18 points)

Copy the DM implementation to a file named `DM+NPP.py`. Within `DM+NPP.py`, implement a simulation of the NPP. Assume that each job requires at most one resource and that all CPU-only computation is completed before the CPU+Resource execution. Before you begin, consider how to track if all the CPU-only computation has completed.

1. Based on the DM implementation (whether the provided code or your code from HW3) and the NPP, which portion of the python script will need to change to track resources and implement the NPP? Explain what you will change and why in `README.txt`.
2. Implement the DM+NPP following the assumptions stated above about when resources should be required. Name your implementation `DM+NPP.py`.

Grading:

- (8 points) Explanation in `README.txt` of what to change and why.
- (10 points) Implementation of `DM+NPP.py`.

## C - Priority Inheritance Protocol (20 points)

Implement a simulation of the PIP with DM in a file named `DM+PIP.py`. Assume that each job requires at most one resource and that all CPU-only computation is completed before the CPU+Resource execution. Explain what parts of the code need to change between the DM+NPP and DM+PIP implementations and why in `README.txt`.

Grading:

- (10 points) Explanation in `README.txt` of what to change and why.
- (10 points) Implementation of `DM+PIP.py`.

## D - Priority Ceiling Protocol (18 points)

Create a new sample task set in the file `test3.json` with four tasks that shows at least two differences in execution with the PCP instead of the PIP. Show the task set scheduled by DM+PIP and DM+PCP. Your scheduler does not need to be able to simulate the DM+PCP. Your schedule for DM+PCP may be drawn by hand (and then uploaded) or by using the visualizer.

Grading:

- (2 points) Readable and correct formatting of JSON.
- (4 points) A description in `README.txt` of how the schedules ought to be different. Point out at least three executions that differ between the protocols.
- (4 points) The tasks themselves, that they follow the instructions (show a job completion for each task, etc.) and should cause the described differences.
- (2 points) Illustration of task set with DM+PIP in `test3_DM+PIP.pdf`.
- (6 points) Illustration of task set with DM+PCP in `test3_DM+PCP.pdf`.

## E - Testing (12 points)

1. Test your synchronization protocol simulations with the task sets that you developed in the first part of this assignment. Produce an image for each schedule, naming it to match the correct file listed below:

- `test1_DM.pdf`
- `test1_DM+NPP.pdf`
- `test2_DM+NPP.pdf`
- `test2_DM+PIP.pdf`

2. Look at these files. Do they produce the differences you expected and noted in Part A? If not, find what went wrong and update either your task set or your implementation.

Grading:

- (4 points) Each file exists and shows a schedule.
- (8 points) Schedules reflect the descriptions in `README.txt`.

# Part 3: Submitting your assignment.

Your assignment should be in `/home/<yourCSlogin>/HW4`. I will collect homework from there and check the last time that each file was modified. This assignment is due on Nov. 21, 2018 at 9:05am EST. If the files have been edited after then, I will assume you have chosen to use one or more late days. Make sure that the following files are in `/home/<yourCSlogin>/HW4`, especially if you were working in a different directory or on a different machine:

- `test1.json`
- `test2.json`
- `test3.json`
- `README.txt`
- `DM+NPP.py`
- `DM+PIP.py`
- `test1_DM.pdf`
- `test1_DM+NPP.pdf`
- `test2_DM+NPP.pdf`
- `test2_DM+PIP.pdf`
- `test3_DM+PIP.pdf`
- `test3_DM+PCP.pdf`

To grade your Python scripts, I will use your tests along with other tests. I will look at each implementation to award partial credit, so make sure your code is readable.