CS 241 Section Week #6
(09/29/11)

MP #4

MP4 Forward

In MP4, you will add code to a simulator for a CPU scheduler.

- We provide you with the code for the simulator.
  - You don't need to understand this code to understand this MP.
  - You should consider the simulator a 'black box'
- You need to implement these algorithms:
  - fcfs: First Come First Serve
  - pri: Priority Scheduling
  - ppri: Preemptive Priority Scheduling
  - sjf: Shortest Job First
  - psjf: Preemtive Shortest Job First (by Remaining Time)
  - rr#: Round Robin

MP4 Forward

- Every modern scheduler uses a priority queue to prioritize what task to run next.

- [Part 1] requires you to implement a priority queue library, **libpriqueue**.

MP4 Forward

- libpriqueue contains nine required functions:
  - State-related functions:
    - priqueue_init(), priqueue_destroy()
    - priqueue_size()
  - Adding and removing elements:
    - priqueue_offer()
    - priqueue_remove(), priqueue_remove_at()
  - Accessing elements:
    - priqueue_peek(), priqueue_poll()
    - priqueue_at()

MP4 Forward

- The priqueue_init() function takes in a comparer function:

  -
    ```
    void priqueue_init(
        priqueue_t *q,
        int(*comparer)(const void *, const void *)
    )
    ```

- This comprarer function is the same function as **qsort()**.
  - Compares two elements, returns the an int if one element is less than, equal to, or greater than the other element.

- We'll look into programming this later.

MP4 Forward

- You now have a priority queue that can prioritize elements based on any function you program.

- Now, it should be simple to implement a scheduler. In [Part 2], you'll implement a second library: **libscheduler**.

MP4 Forward

- You need to fill in 3 scheduling functions:
  - scheduler_new_job()
  - scheduler_job_finished()
  - scheduler_quantum_expired()

  Note that these are the only times that the scheduler needs to make a decision!

- The scheduler_start_up() and scheduler_clean_up() functions are provided to allow you to initialize your scheduler and clean up any memory used.

MP4 Forward

- You also need to fill in 3 statistics functions:
  - float scheduler_average_response_time()
  - float scheduler_average_wait_time()
  - float scheduler_average_turnaround_time()

    These are called at the end of the simulation.

- We also provide one function debug-related function: scheduler_show_queue().
  - After every call our simulator makes, we'll call this function and you can print out any debugging information you want.

MP4 Forward

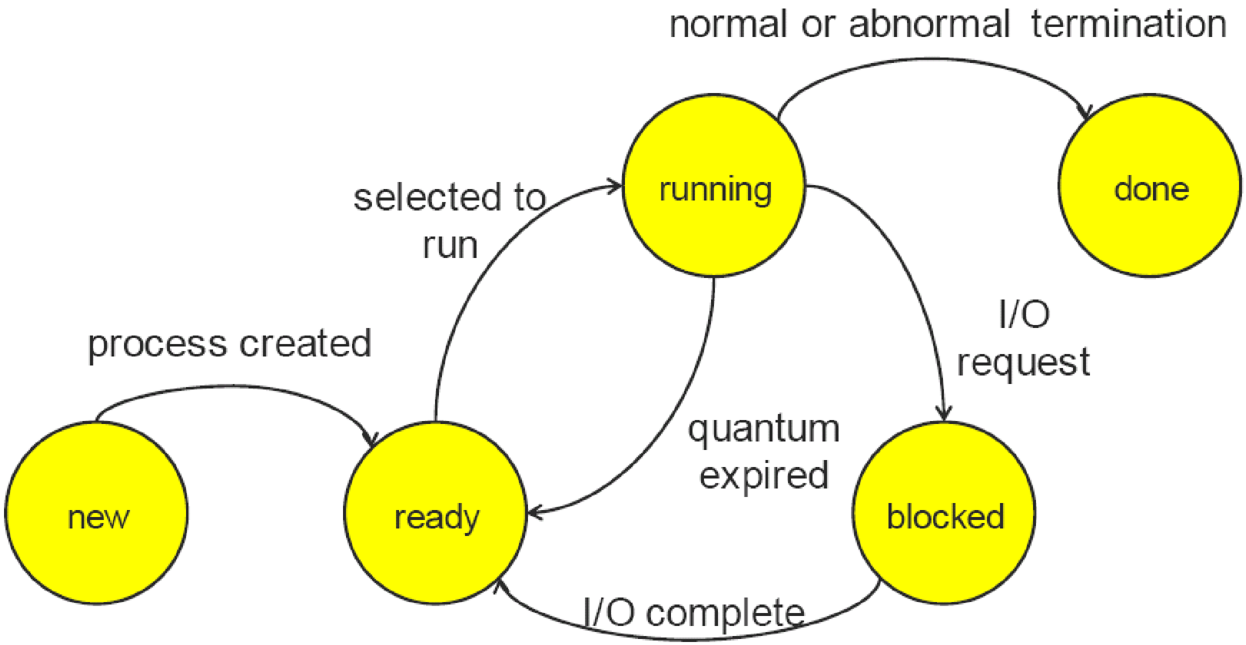
- For success on this MP:
  - We provide queuetest.c, a program to help you test [Part 1] independent of [Part 2].
  - We provide 54 example output files and a program, examples.pl, to run all 54 examples at once and report any errors.

- Requires a good understanding of data structures, scheduling, and pointers all in one MP.

Good luck!

MP4: Relating Back to Lecture…

5-State Model - Transitions



Lets Go Programming…

Programming

- **Question**:
  - What are some things we can do on a char-by-char basis to a string?
  - Ex: Make lowercase letters uppercase.
    - c ➔ C

Programming

- **Question**:
  - What are some things we can do on a char-by-char basis to a string?
  - Ex: Make lowercase letters uppercase.
    - c → C
- **Goal**:
  - Create a program that allows us to manipulate strings in **all** the different ways you described above.
  - …all using one single function with different parameters.

Programming

- **Naïve Solution:**

  -
    ```
    void mainp(char *s, int what_to_do)
    {
          if (i == 0)
              upper_case(s);
          else if (i == 1)
               lower_case(s);
           else if (…)

    }
    ```

- What’s wrong with that?

Programming

- **Lets do better….**
  - **File: ds/ds5/1.c**