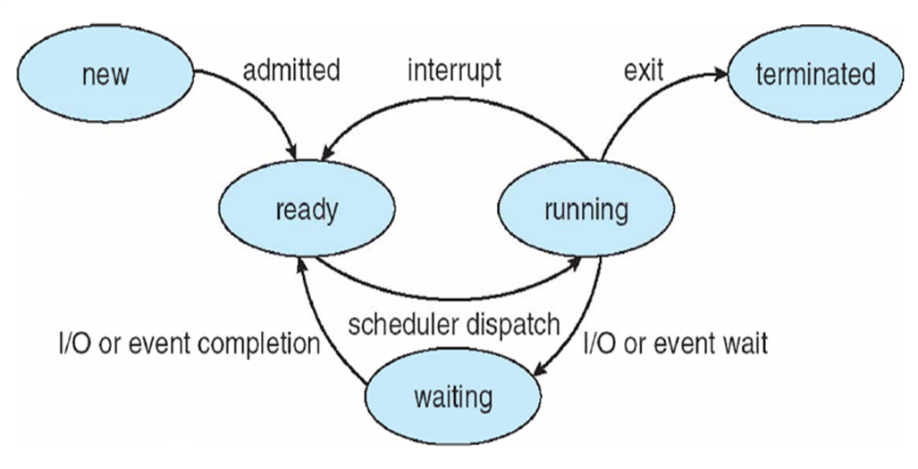
**CSIII: Programming Patterns**

**Lab 11 Assignment:**

**Prototype State Scheduler**

The project is due in one week: by the midnight on the day of the next lab. Make sure to include your name in comments of the source files.



* Using State Design Pattern, implement Process class that provides an interface to the five state process model of operating system scheduler implementation. The process may be in one of the states shown in the figure. The states should implement transitions as shown. Each process should have a unique identifier (**Hint:** maintain a static variable in class Process to keep track of the next available identifier, then increment the identifier once a new process is created.) The states should also implement a reporting function outputting what state the process is in. The process should be able to output:
* process 1 is running

If the transition is not in the appropriate state, for example if schedulerDispatch is invoked on a process that is in running state, no action should be taken (no state transition should occur).

**Hint:** Use the state of Zork example that we studied in class a point of departure for your implementation.

* Use Prototype Design Pattern to create a new process. That is, the new process should be cloned from the prototype. The prototype should be in new state of the five state process model above.

Implement a process ready queue as follows. Use STL queue of processes. The first process in the queue is in running state. Randomly (with probability of 33% for each case), the running process may either exit, be preempted or execute an I/O request.

* + In case of exiting, the process is destroyed.
  + In case of preemption, the process moves to the tail of the queue and becomes ready.
  + In case of I/O request, the process becomes waiting and moves from the running queue to a waiting set. With a certain probability, any process from the waiting set completes the I/O request and returns to the tail of the waiting queue becoming ready again.

Report on the state of processes after each transition.

Demonstrate correct implementation of your ready queue and Prototype by admitting and scheduling 4 new processes.

**Milestone:** Implement appropriate state tansitions for a single process.