NB for Josh: a reference for markdown and in case you're wondering, there's lots of ways to convert markdown to pdf easily

Design Document - Jobs Manager

Brennan McFarland and Joshua Reichman

Overview

Our project consists of a program to monitor and manage the concurrent execution of other programs via command line. The idea is that the user can queue up bash commands in an interactive terminal as "jobs" and the manager will then schedule CPU time to run them concurrently or with the appearance of concurrency if there is only one CPU core. Preference will be given to processes that are queued first, and the user can also assign priority to certain jobs. The processes will be scheduled in a manner similar to the popular telescope scheduling algorithm. Additionally, the user can type a command to display the list of jobs and their status in a manner similar to the "top" command and stop running jobs as with "kill". In a nutshell, the manager function similarly to queuing and viewing the status of batch processes on the HPC cluster.

Files

TODO: add to/change this as we go
* **jobsmanager** - a bash script to start the application
* **jm.py** - runs the jobs manager application; python execution starts here
* **jmshell.py** - contains the shell specification (a class inheriting from the cmd.Cmd class in the python standard library) that binds commands to program functions
* **jmmanager.py** - handles the jobs themselves, keeping track of them and handling the calls bound by jmshell to create, kill and list new jobs TODO: may need to split this up
* **jmjob.py** - a job and relevant information pertaining to id (ID, PID, etc)

Data Structures

TODO: this section

Usage and Sample Output

- **runjob** ***bash command*** - add a job to the queue

```
\>runjob echo "hello!"
hello!
```

- **runjob -p** ***priority bash command*** - adds a job to the queue with the specified priority (an integer), with 1 being the highest priority

```
\>runjob -p 1 echo "hello!"
hello!
```

- **lsjobs** - display the list of jobs and their statuses and IDs (specific to the jobs manager, not the same as PIDs) TODO: update the example below

```
\>lsjobs
ID      PID     STAT     NAME      TIME      %CPU
0001   27129     R        sleep.c   0:00:10     0
```

- **killjob** ***job*** - cancel the specified job, which can be either the job's ID or name, with undefined behavior for which job of the same name is killed in the case of multiple of the same jobs

```
\>killjob 0001
killing job 0001: sleep.c
```