Distributed Computing Exercise Using HTCondor on CHTC Machines

Note that in this exercise we are using small data for which the parallel step slows us down! Working in parallel will help when we have more data than fit in memory on one machine. The goal here is to learn HTCondor scripting to prepare for a large search for Lyman-break galaxies.

- Write a HTCondor DAGMan script, `words.dag`, to make a sorted list of the words and their counts in the works of Shakespeare from `shakespeare.tar` (which extracts to a `shakespeare` directory). Include the following steps:
  - Write a PRE "`.sh`" script:
    - Write all the plays to one large file.
      - `wget` can download `shakespeare.tar`.
      - `tar` can extract it.
      - `cat` with wildcards can concatenate all the plays into one file.
    - Break the large file into 5 smaller files.
      - `split` can do it. See `man split` and the data splitting example in the standard deviation program from lecture.
  - Write a HTCondor "`.sub`" script to process the 5 files in parallel by calling a "`.sh`" script on each file:
    - Reformat the current file so it has one word per line. This may require replacing each tab with a space and then replacing one or more spaces with a newline.
      - `sed -e 's/PATTERN/REPLACEMENT/g'` ("stream editor: substitute `PATTERN` with `REPLACEMENT` globally") can do it.
    - Sort the file of words.
      - `sort` can do it.
  - Write a POST "`.sh`" script:
    - Merge the sorted small files into one large sorted file.
      - `sort` can do it. (Don't just run `sort`, which would waste your earlier parallel sorting of the small files. Instead, look in the `sort` manual page to see how to *merge* several sorted files *without* sorting. See also the Wikipedia page for k-way merge.)
        Note: `sort` can produce output that looks unsorted because it relies on your computer's locale settings. You can ensure familiar English sort order by running `export LC_ALL=C` in your script before running `sort`.
    - Convert the large sorted file to one called `countsOfWords` whose lines have the form `count word`
      - `uniq` can do it.

Submit one tar file per group, `words.tar`, that extracts to make a directory `words` containing:

- your `words.dag` file and any other code you write

- your `countsOfWords` file
- a plain-text file `README` that includes information on your group members (1 to 4 students) in the line format `NetID,LastName,FirstName`. For example, if Wilma Flintstone (NetID: wflint3) and Charlie Brown (NetID: cbrown71) worked together, their `REAMDME` file would be:

  ```
  wflint3,Flinstone,Wilma
  cbrown71,Brown,Charlie
  ```

- no input data