Treader – A GUI-based MP3 Tag Reader

Heather Hoaglund-Biron

Modules used:

* Databases: SQLite
* Multi-processing: multiprocessing
* MP3 tag reading: mutagen (http://mutagen.readthedocs.org/)
* GUI: Tkinter

Design:

The main file (application.py) contains an Application class. An Application is a subclass of Frame, a Tkinter object. An Application contains a references to both a DatabaseHelper object (defined in dbhelper.py) and a TagReader object (defined in tagreader.py). The TagReader uses the Mutagen package to read metadata from specified MP3 files and returns them in a dictionary. The DatabaseHelper is the interface to the SQLite database, reading and writing MP3 tags to the database.

Reading tags and writing them to the database is done in a different process than reading tags from the database. The GUI also runs in a separate process. I’m still unsure whether the GUI should run in the same process as reading tags from the database, as the GUI will need to read tags from the database to update itself, and the extra separation may not be necessary.

The database will have one table, songs. The columns will be title, album, artist, track, and length. The title and album columns will be a combined primary key. Artist, track, and length can be null. All columns are text except for the track, which is an integer, and specifies the track number on that song’s album.

To start up the GUI:

* Run startxwin in the Cygwin command-line
* Run python ./python-300/project/application.py in the X window

Wireframe:



Multiprocess communication:

When the database writer finishes writing one row to the database it will add [something] to the Queue to alert the database reader that it should grab the next row.

To Do:

* Get all applicable tags from mp3 files (now writing the format function for time length)
* Test finished tag reader
* Make tag reader and db helper work together in a test program (read tags, write, then read, in test\_readwrite.py)
* Add track number to application
* Copy test program into application and have it print the information to the screen on “import”, sequentially writing then reading
* Involve multiprocessing