15-395 Lab 5
Persistent Database – Extensible Hashing

Objective

This lab serves to strengthen your coding skills while building an understanding of external data structures. It also gives you the opportunity to turn the wheel once and build and deploy a shared library, an important form of real-world software components.

Disk-based metadata will be important should you take databases. But, it is also important in, for example, operating systems, where the file system consists of external metadata. The database that you build for this lab will serve as a platform for the next lab which will introduce you to Berkley sockets as well as concurrency.

Times of Interest

Assigned: Tuesday, October 30th, 2007
Due: Thursday, November 8th, 2007

Part 1: Background Reading

The structure of your database should be an extensible hash. The following survey paper is a good scholarly source for information, which also contain citations to the papers originally presenting the various techniques. It is available via the ACM's digital library. Of course, google.com can turn up information, of all kinds and quality, wholesale.

- Enbody, R.J. and Du, H.C., "Dynamic Hashing Schemes", ACM *Computing Surveys,* vol 20, no 2 (June 1988): 85-113:

Part 2: Choose Your Poison

There are several different schemes for external hashing. Pick one. My personal favorite is the *Linear Hash (LH).* But there are plenty of good reasons to select others. You are not required to implement deletes beyond tombstones. But, please do feel free, to shrink the data structures, if you'd like.

Part 3: The API

Your solutions should have two parts. One part should be a shared library presenting a simple API. The API should include *open*, *close*, *put*, *get*, and *remove*.

The library should be able to support multiple databases simultaneously. In other words, it should be session-oriented. A session begins with some type of "open" and ends with some type of "close". The open associates the database with a handle. This handle, in turn, is provided to the other functions, including the close, to identify the particular database. At this point in time, you need not worry about multiple concurrent sessions of the same database. We'll introduce that problem for the next assignment.

You might want to take a look at ndbm or gdbm ("man ndbm", "man gdbm") to get some ideas about how to structure your API. In particular, you might want to pay careful attention to how they represent the data and the key to the database.

Part 4:Storage

Your databases must be a persistent, external database. If you are keeping it in memory – you need to be able to live without it.

For those with more experience (or just dumb ambition), please consider making your lives a bit more complicated. Consider implementing caching within the shared libraries global space. It is sticky, but can be fun.

Part 5: Common Sense

Your project must include a *Makefile*. It should be well-written, including comments, descriptive identifiers, &c

Your database files will contain persistent information. When things don't work right, it can be examined to help identify the problem *od*, "octal dump" ("man od") is an excellent tool for viewing raw binary files.

But, you might want to construct some tools that allow you to do more. For example, you might want to build some tools that know the specific structure of your database and can do some of the repetitive parsing for you.

Last, but not least, you'll want a ready source of data with which to feed your database. Try finding some on the Web and mining and formatting it using shell scripts. You want to test with thousands of records – or, well, even way more.