# ECE 656 Fall 2020: Sample Project Proposals

# Un*x File-System Replacement

Homer Simpson
homer@theSimpsons.com
20123456

Bart Simpson
bart@theBartMan.com
66666666

The standard un*x file system is very inefficient for a number of reasons. However, the basic directory structure is well understood by users and developers alike, as are the various filesystem utilities (ls, find, grep, *etc.*). Using a relational database as a file system should be much more efficient, though only if we do so in a non-standard way. The standard method is to implement the various filesystem system calls (open, close, read, write, lseek, *etc.*) for the new file system and then nothing else needs to change for the user (existing utilities and shell remain unaltered). The problem with this is that doing so would essentially negate any advantage that the use of a relational database would bring. Instead of doing this, it is necessary to rewrite the standard utilities so as to provide the standard "look-and-feel" of that utility, but doing so on top of a relational database.

For our proposed project we expect to do the following:

1. Our data source will be the Ubuntu filesystem files that we are using for our relational database. We expect to add some additional files from our own home directories to add some file types that are not typically present in the base Ubuntu filesystem.
2. Create an entity-relationship model for our un*x filesystem. We will include all relevant:
   - entities: files, directories, soft links, hard links, *etc.*
   - attributes: file name, size, owenership, permission bits, *etc.*
   - relationships: file-to-directory; directory-to-directory, *etc.*
   - cardinality constraints and participation constraints: TBD
3. Create the necessary relational database from our ER model, appropriately normalized. We expect to use the type BLOB (Binary Large OBject) or TEXT types, as appropriate for files. We will write the necessary SQL to create all necessary tables, keys, *etc.*
4. Populate our database with appropriate sample data; our starting point would be to write a small utility that would copy the file system from a un*x box into our system.
5. Create the following client utilities to run on top of our database:
   (a) rdbsh: a shell program that can, at a minimum, keep track of the current working directory, change the working directory (cd), maintain a PATH variable, and execute any executable program that is in the PATH.
   (b) ls: it will be able to accept the "-l" argument or just plain usage; we may implement additional ls options. The ls command will be able to execute within rdbsh or as a standalone program.
   (c) find: we will implement a subset of the un*x find utility that will accept a directory name and (partial) name of the file being found. We will not do full regex pattern matching on the filename but simply assume portion of the desired filename. The output will be the "ls -l" results for all matching files. The find command will be able to execute within rdbsh or as a standalone program.
   (d) grep: time permitting, we will implement a simplified form of grep, which will accept the (partial) name of a file and seek the relevant pattern in the matching file(s). We will output the line number and line for the matching lines.
6. We do not yet know what or how we can data mine the filesystem but expect to figure that out as we go along.

# A Simple Social Network

Maggie Simpson
maggie@awesomeBaby.com
77777777

I plan to create a simple social network and populate it with data that I will scape from publicly available Twitter postings and accounts. Broadly speaking, all social networks are centered around "people" posting "things" about "topics" and commenting on those posts. People have various attributes that may be genetic (*e.g.,* birthdate, sex, *etc.*) or chosen (*e.g.,* vocation, religion, *etc.*). People are part of one or more groups (reciprocal) or follow one or more other people (non-reciprocal) or topics. While Twitter only has the "follow-person" concept, if two users are mutually following each other, I will form them into a group; if a person reads a post on some topic the client I devise will allow the user to follow that topic.

Topics are self-organized by the people in the social network and I will plan to use the Twitter hashtags as identifying topics for postings. I hope to extract subtopics from topics based on creating a topic-tree but that may be beyond the scope of the project.

Things posted are either some initial post, comprising text, image(s), and link(s) by a person on one or more topics. Those who follow that topic or posting person will be notified of the posting, and may choose to respond to the post, either simply with a thumbs up/down or by posting their own text, image(s) and link(s) in response, which in turn may generate responses. Likewise, a person or topic may be searched, and the person receiving the results may choose to respond to those results. If a person has read a post, the system should know this, although "read" may simply mean "the client has presented it to the user."

To satisfy the project proposal requirements I will:

1. use Twitter as my dataset source
2. Build a simple social network and populate it with the Twitter data
3. create a simple command-line social-network client to run on top of our database. This client will allow the user to
   (a) post an initial post on a topic
   (b) follow another person
   (c) follow a topic
   (d) read unread posts since last use
   (e) not have to see the same post twice
   (f) search through posts, read or unread
   (g) respond to a post with thumbs up/down and/or a response post.
   (h) I will add additional items if time permits
4. Data mine as appropriate; I do not know what to do here yet