William Blinn (wb169@columbia.edu)
Jeffrey Lin (jlin@columbia.edu)
Nabil Shahid (ns602@columbia.edu)
Matt Anderegg (mj105@columbia.edu)

Mr. Proper™

Introduction

In this Internet age, people do a lot of chatting. It is not uncommon for one to amass many tens of megabytes of log files in the course of chatting over a month. Sometimes, going through and reviewing these enormously massively gigantic excessive log files can be a daunting task that will bring fear to all but the most chronic of procrastinators. Our language facilitates simple manipulation of said gargantuan log files. Such tasks as converting log files from one logger format to another will be made trivial by implementing them at the language level, hiding its complexity from users. Dealing with entire log files as primitive data types allows users of the language to manipulate logs with the ease of integers and strings.

Mr. Proper™: An easy to use, powerful, multi chat client supporting, log file manipulation language.

Easy to use

Mr. Proper™ hides the complexity of dealing with unwieldy log files from the user, allowing them to handle them the same way that they would handle data types such as integers in other languages.

Powerful

Broad functionality includes analysis of conversations to determine likely personality traits of chat client users, based on characteristics such as text style, writing style, grammar style, and sentence lengths.

Multi chat client supporting

Mr. Proper™ supports log file formats for several of the major chat clients, such as Trillian, AIM, Fire, and gAIM. Converting from one log format to another is as simple as writing a few lines of code, rather than creating an entire program to parse and convert radically different logging formats.

Log file manipulation

At its core, Mr. Proper™ is designed to make operations that users may want to perform on their log files easy. Users can write simple powerful programs in few lines of code, without dealing with such overheads as opening files, reading individual lines, and parsing the text from them. Searching for a particular word in a particular log file is possible with a minimal amount of effort, using Mr. Proper™.

Language functionality:

- Using scoring based on regular expressions (a la SpamAssassin) to determine personality traits of people with whom you speak. Loops would

be used to go through multiple IM transcripts.

- Compiling lists of links that have been sent to you over IM so that you have a centralized list (a further use of regular expressions). This will be accomplished by looping through all IM transcripts for a particular chat client user.
- Converting logs from one program's format to another and the ability to support additional log formats.
- Showing the frequency of conversations based on time-stamps and using it to answer questions such as whom someone talks to most at night or what time of day a particular chat client user chats most.
- Word counting, which would make use of arrays to store most frequently used words and looping to go through several IM transcripts.
- Primitive data types for messages, which contain data about the chat client user, timestamp, date and the text of the message. There will be easy access to all of these primitives simply by opening a log file.

Why is it useful?

- Have you ever wondered whom you talk dirty to the most? Have you ever sat there at your computer and wondered how many times you said "fuck" yesterday?
- Find links that friends sent to you.
- Search logs for certain things.
- Rate the intelligence level of your friends

- Analyze conversations for personality traits and talking styles
- When you switch chat clients you can easily convert your old conversation logs into the appropriate format for your new chat client in order to preserve your conversation history without giving up consistency.
- Parse and format conversation logs to make online publishing of conversation snippets a breeze. Conversation bits may be automatically reviewed with regular expression scoring to flag questionable or deeply personal content that you might not want put online.

```
// Primitive data types:
// Individual log file types, e.g. Trillian, AIM, Fire, etc.
// Log file primitives would act like arrays, with each spot in
// the array being a line of the conversation
// Attributes: text, speaker, time, date

// So to declare a new log file for the Trillian format:
trillianlog temp;

// Then to assign to this variable the log file for a screen
// name called mrproper:
temp = open(mrproper);

// for format conversion
trillianlog tlog = open(mrproper);
aimlog alog;
alog = (aimlog) tlog;    // by cast
     // or
alog = toaim(tlog);      // by explicit method

// To write the log back to file in the format specified by the
// data type. This also clears the variable so it can't be
// modified or used anymore. "close" in this case is a reserved
// word in the language.

close(alog);

// Operations on the log data types could include:
log1 + log2;             // concatenation of the two log files,
                         // with log2 being concatenated to the
                         // end of log1
log1 - <screenname>;     // remove everything said by the
                         // screenname indicated by <screenname>

// Sample Program 1
// I want to open a Trillian log, and an aim log, concatenate
// them, and save them as a Fire log.
// start
trillianlog logt = open(mrproper);
aimlog loga = open(yourbytes);
firelog logf;
logf = (firelog) (logt + loga);
     // or
logf = tofire(logt + loga);

close(logf);
// end
```

```
// Sample Program 2
// Print out the conversation spoken by mrproper in the first 20
// lines
// start
trillianlog logt = open(mrproper);
for (int i = 0; i < 20; i++) {
    if (logt[i].speaker == mrproper) {
       print(logt[i].text);
    }
}
close(logt);
// end
```