gscc
A General Search and Compare Compiler

gscc is a text manipulation language that rivals existing programmatic solutions. It is compact, intuitive and lightweight, giving programmers a means to quickly manipulate their text-based targets.

Eric [G]arrido
Russel [S]antillanes
Casey [C]allendrello
Ho Yin [C]heng

An Introduction to gscc

gscc is a simple text processing language which provides facilities to execute a variety of rich text manipulation commands on a text file.

Background

The wide-range usage of computers in all industries has necessitated employing administrators who are skilled in maintaining the systems on which valuable work is done. Typically, the configuration of these systems relies on complex text files, which must often be manipulated by the administrators. With a computer at their disposal, the administrators created ways to make the machine perform the complex pattern-matching and text-replacement operations.

As computers have been in place for several decades, the needs and abilities of users have increased. Today, very complex and useful systems are maintained in text files: statistics, programming language code, academic papers, news articles, address books, etc. Facilities must exist to modify these types of files based on a set of rules, defined by the user, and yet still be easily accessible to the user.

Goals of gscc

gscc seeks to be a programming language that fills this niche. It will provide the user a small set of very powerful utilities to allow complex manipulations of their text files.

Simplicity

gscc should be accessible to a wide-range of people with varying computer skills. Therefore, gscc uses simple English commands to manipulate text in complex ways.

Streamlined Operation

gscc iterates over each line and performs all defined commands on that line sequentially, only passing through each file once.

Portability

Since gscc is implemented in Java and uses the Java Virtual Machine to operate, gscc is by definition platform independent. The user is able to manipulate text the same way on a Windows platform as he is a Unix platform.

Language Features

Small, powerful command set

gscc will present a small number of commands to the programmer that will allow for huge possibilities of utility. The commands allow the programmer to perform the few basic text manipulations: insertion, deletion, appending, and searching. The programmer can perform these operations based on regular expressions and other logic-based decisions.

User-defined subroutines

As text processing requirements vary, the user will be able to specify domain-specific subroutines to acquire the desired result. The user can extend the built-in capabilities of the language by creating subroutines. These subroutines also allow for high amounts of code reuse and simplify potentially-complex code.

Unicode Support

As with international business, the need for manipulating foreign character sets continually increases. By virtue of being based on Java technology, gscc

gscc in Practice

A First Syntax Example

The following example replaces all instances of “Hello” or “hello” with “Goodbye!”

```
set %bye = "Goodbye!";
find /[Hh]ello/ global {
        set $Match %bye;
}
```

Further Examples

The following example prepends all lines that match this regular expression with the line length, but deletes them if they don't begin with A.

```
find /t?[a-z]*66/ line {
        if( $Line.char[0] != 'A') {
                del $Line;
        }
```

```
        # the last block can also be implemented as:
        # if( match /^A/ $Line ) {
        #        del $Line;
        # }
        else {
                insert $Line 0 $Line.length
        }
}
```

Imagine a file of grades, where you want to replace numbers 90 through 93 with A-:

```
find / 9[0-3] / global {
        set $Match "A-"
}
```

…except you have a problem: the University's mainframe system was written in FORTRAN and can't handle last and first names longer than 15 characters. You want to truncate each tuple to "lastname firstname grade".

```
find /^[A-Za-z] / line {
        while( $Matches.match[0].length > 15){
                del $Matches.match[0] 16;
        }
        append $Matches.match[0] ' ';

        while( $Matches.match[1].length > 15){
                del $Matches.match[1] 16;
        }
}
```