SCRIPT-EDIT
A language that will let you edit and script simultaneously.

BETHANY SOULE
bms2126@columbia.edu

BHAVESH PATIRA
bp2214@columbia.edu

DENI PEJANOVIC
dp2232@columbia.edu

MARC VINYES
mv2258@columbia.edu

PROJECT PROPOSAL

Description:

ScriptEdit is a language that allows you to automatically generate text from a limited set of instructions. You can write new files with this text, or insert it into an existing file.The instructions may use the text of the edited file, external files or text generated by other standard input/output based applications as input. ScriptEdit is similar to a macro processor in the sense that replaces text with other text, but it can also create new files from one single source.

Motivations and Goals:

The main goal of this language is to allow the user to edit files and the script operations that are needed to create their content all from within one single source file. Often, editing content text files (HTML, Latex, XML, etc) is a process that involves several different steps and programs - like separate bash scripts, a text editor, and other console programs (e.g. using ImageMagick to edit images or using Matlab to create graphs that will be linked). ScriptEdit is a way to put all those different process calls together with the content text file.

Some advantages of scripting operations within the source content file:

- **Visualization** Contents and related scripts are written in the same area so it's easy to check that they are coherent using standard editors.
- **Programs that need to be run before writing contents are executed automatically and you don't have to worry about the right order**. If a long set of programs have to be executed each time, normally a shell script grouping them is created and then executed, but then new users that edit the file should be aware that this file should be executed and will have to figure out if the script has to be run before or after editing some files. Sometimes, the process can be tedious and hard to reproduce like run program1, edit file1, run program2, edit file2 . So ScriptEdit eases this task by running the scripts and outputting the text at the same time in the right order. Moreover modifications of the format of the input file are also more easily synchronized with the script.
- **Templates and macros are easy**. Some languages like C already have a good preprocessor that allows using templates " #include, #ifdef, etc -, but others like HTML or XML don't, and users are asking for them. Macros are normally placed in the editor but each editor uses different implementations of them, so you have to learn multiple ways to do the same.

Other languages have already been built to solve this sort of problem. In the particular case of processing HTML text files: HTP(http://htp.sourceforge.net/) and HPP (http://citeseer.ist.psu.edu/douglis97hpp.html) are good examples , but they are more specialized than what we have in mind. Perhaps a more comparable existing language is M4 (www.gnu.or/software/m4).

Overview of its syntax:

The syntax of ScriptEdit borrows from the C preprocessor and bash.

Variables are identified “${variable}", as per bash.

Instructions:

- Begin with #, (as per the C preprocessor) followed by optional parenthesis containing a list of the arguments
- Input text is specified between brackets ‘{‘,’}‘ or by adding ‘0’ or ‘1’ prefix to the instruction name as follows:

```
#0instruction(argument)
input text
#1instruction(argument)
```

is the same as:

```
#instruction(argument)
{input text}
```

Instructions can be inserted in the middle of a text file, without any additional formatting and when the file is compiled its instructions are executed and their output is inserted at the line where they are located within the file. A simple example follows:

Example 1:

```
#set(name){Carla}
#set(place){Barcelona}
${name} was born in ${place}
```

Would output:

```
Carla was born in Barcelona
```

Example 2 (resizing pictures, writing translations of documents with lots of structure, generating HTML file)
In this example a personal website displaying all the pictures of its home folder is produced in two languages: Spanish and English.

```
#foreach
      #(la=){es}
      #(color=)(#FFFFFF)
#next
      #(la=){en}
      #(color=)(#FF0000)
#0do
      #bwrite(index_${la}.html)
            #set(${la})

            <html>
            <body background="#${color}">
            #include(header.sehtml)
            <table>
            <tr>
            <td><img src="icon.gif"></td>
            <td>
            #ifdef(es){Bienvenidos a mi pagina personal}
            #ifdef(en){Welcome to my personal website}
            </td>
            </tr>
            </table>

            #ifdef(es){Fotos:}
            #ifdef(en){Pictures:}
            <br>

            #foreach(pic)
                  #exec(ls "l *.jpg)
            #do
                  #exec(convert ${pic} "resize 400x20 "o o${pic})
                  <img src="o${pic}">

            #include(footer.sehtml)
            </body>
            </html>

            #unset(${la})
      #ewrite(index_${la}.html)
#1do
```

hence, the file index_en.html will be generated as follows:

```
<html>
<body>
<tr>

<!--header begins -->
<table id=header>
<td>Home</td>
<td>Projects</td>
<td>Contact me</td>
</tr>
</table>
<!--header ends -->

<table>
<tr>
<td><img src="icon.gif"></td>
<td>
Welcome to my personal website
</td>
</tr>
</table>

Photos:
<br>

<img src="pic11-02.jpg">
<img src="pic11-01.jpg">
<img src="pic02-21.jpg">
<img src="pic03-32.jpg">

<!--footer begins -->
<br> (C) M, Barcelona, 2006
<!--footer ends-->

</body>
</html>
```