Project Proposal

Windshield

Windows Shell Script

Columbia University
COMS W4115 Programming Languages and Translators
Spring 2007

Prof. Stephen A. Edwards

Team members

Wei-Yun Ma	**wm2174**	**wm2174@columbia.edu**
Tony Wang	**tw2174**	**tw2174@columbia.edu**
Tzu-Jung Liu	**tl2263**	**tl2263@columbia.edu**

Introduction

On command line-based operating systems, shell scripts are especially useful. Through the evolution of versions, shell programming has been consistently integrating new functions, such as variables, loops, conditions, strengthening shell scripts to be capable of much more. One of the important features of shell scripts is its ability to “glue” software, which is available under its current operating system (shell scripts are also known as glue languages). Due to the rise of GUI, WWW, and component frameworks, we need a better, more powerful alternative to integrate software.

With the likes of Bourne Shell, Bourne-Again Shell(bash), Korne Shell, UNIX had enjoyed a great variety of powerful shell scripting language. Windows on the other hand, utilizes batch files and software such as 4DOS, 4NT to strengthen the former; however, still can not program as powerfully as UNIX shell scripting language.

We therefore propose to implement Windows Shell Script(Windshield). Using the Bash script as reference and implementing the fundamentals, we try to develop a prototype of a Windows Shell Script and a corresponding compiler to simulate the functions of the interpreter. We also seek to add other useful functionalities to further enhance programming with Windshield. Our compiler will read in the source code, and produce a Perl file, executable under Windows.

Variables

Like the Bash shell script, there are no specific data types in Windshield. A variable in Windshield can contain a number, a character, a string of characters. Programmers can save the work of declaring variables, instead assigning values to its reference creates it.

Sample:

```
STR="Hello World!"
echo $STR
```

Line 2 creates a variable named STR and assigns the string "Hello World!" to it. Then the VALUE of this variable is retrieved by putting the '$' in at the beginning.

Condition and Loop Statement

Windshield provides the basic if-then-else condition statement and three loop statements: while, for and for in. The syntax is demonstrated below:

if [condition];
then
  code if condition is true.
else
  code if condition is flase.
fi
	while [condition]
	do
	  code if condition is true.
	done

for [expr1; condition; expr2]
do
  code if condition is true.
done
	for variable in list
	do
	  code
	done

Using the command line or calling other programs

One of the most important features of a shell script is it can directly utilize shell command or call existing programs to collaboratively achieve the goal.

Sample:

cat file1 | sed s/Hello/World/ > file2

C:\myprogram.exe file2 > file3

copy the content into file2 but replace any “Hello” with “World” in file2.

Execute *C:*\myprogram.exe which take the inputfile-file2 and output an outputfile- file3

Unix/Linux shells provide much stronger commands than Windows. Therefore in Windshield we do not adopt/allow DOS commands. On the contrary, the small part but basic Unix/Linux commands will be included in Windshield, such as echo, cp, mv, rm, mkdir, rmdir, cd, df, du, ls, cat, pwd, basename, expr and read. We also intend to implement two useful utilities-sed and grep in Windshield. The two utilities allow regular expression representation for their inputs. Therefore, we would compile regular expression statement into their corresponding target codes.

I/O Redirection

I/O redirectors are used to send output of command to file or to read input from file. Windshield provides basic I/O redirection functions which used frequently in Unix/Linux shells. There are three main redirection symbols in UNIX/Linux shell: **>,>>,<.** Windshield would provide two of them: >, >>.

(1) >

Syntax:

command/program > filename

To output command/program result (output of command/program) to file. If file already exist, it will be overwritten else new file is created. Ex: *$ ls > myfile*

(2) >>

Syntax:

command/program >> filename

To output command/program result (output of command/program) to END of file. If file exists, it will be opened and new information/data will be written to END of file, without losing previous information/data. And if file does not exist, then new file is created. Ex: *$ date >> myfile*

Pipeline

Pipeline lets programs use the output of a program as the input of another one with the symbol: |

Take the previous example:

cat file1 | sed s/Hello/Word/ > file2
myprogram.exe file2 > file3

If file2 is not the concerning file, the above two lines can be combined into the following one line:

cat file1 | sed s/Hello/Word/ | myprogram.exe > file3

Compile Example

The following program is to convert the filename extension “.htm” of every file into “.html”

Source Windshield Code:

```
for file in *.htm;
do
    echo "processing $file"
    mv $file `echo $file | sed s/\.htm/\.html/`
done
```

Target Perl Code:

```
foreach my $file (glob "*.htm")
{
    print "processing $file";
    $desfile=$file;
    $desfile=~s/\.htm/\.html/;
    rename $file $desfile;
}
```