

## WELSH: WINDOWS EASY LANGUAGE SHELL

Lane Hill, Southwestern University

Faculty Advisor: Dr. Suzanne Buchele, Southwestern University

### ABSTRACT

The Windows Easy Language Shell, WELSH, is a command-line shell for Windows that is an alternative to the standard DOS shell that comes with Windows. WELSH's goal is to provide an easy but powerful command-line shell for users, to help bridge the gap between similar Linux and Windows shell commands and to educate users on the benefits of using a command-line shell.

### INTRODUCTION

A shell is a piece of software that allows a user to interact with a computer's operating system. There are two types of shells: graphical shells and command-line shells. Most users are more familiar with the graphical shell (such as Windows Explorer or the Gnome Desktop Environment), and it has been shown that these shells are easier to use for most users [2]. As a result, the concept of a command-line shell that operates mostly on text commands can be seen as "scary" or more advanced by users who do not know much about computers [3]. However, it is possible to get more tasks done in the time it takes to type a command in the command-line shell than in a graphical shell. Also, the use of a command line allows the user to focus on one task at a time, while still allowing commands to be processed in the background. Furthermore, the user can interact with the operating system at a deeper level of detail than what may be offered with the GUI [3,6].

However, there is the problem of differing shells, usually in commands or syntax. Windows' cmd.exe, for example, is a DOS command interpreter for Windows. This means that it only can understand DOS syntax and vocabulary, which can be different from the UNIX syntax and vocabulary [4]. This shell - as well as the only other major Windows shell, Microsoft's PowerShell (which is only standard on Windows Server 2008 and future Windows 7 releases) - uses a similar but still different set of vocabulary than modern shells on Linux [5].

Another issue with command-line shells is that the user has to deal with a path variable - that is, a variable that is a list of paths that the shell can access routinely used programs in. This way, the user only has to type in the name of the program to execute the program, instead of a full path to the program. The problem is that these path variables can be confusing for new users, and there is not an easy, user-friendly way to change these variables [1].

Also, the language of the user can be a problem as well. For example, in English, `tracert` is a Windows program that is an amalgamation of the words "trace" and "route." However, in French, the command isn't `retroouvercn` - which would be an amalgamation of "retroouver" and "chemin." For non-native speakers, this can be confusing and takes away an element of user-friendliness.

The last problem is the use of archaic commands. For instance, how would a new user know that `ls` will list out the files in the directory? Or that `rm` deletes a file and, if used

incorrectly, can easily wipe out an entire folder of important documents just because the user supplied the wrong argument?

WELSH is intended to make it easier for Windows users to become adjusted to a command line shell by making the experience enjoyable and making it possible to get tasks done quickly. Furthermore, the commands are simple and to the point and are aliased so that more than one name for the command points to a single command. It is also possible for a user to create aliases for a command or a file. This helps with the path variable issues. For instance, to access the web browser Firefox in such a way that the user just has to type in "firefox" in a command line (without navigating to the directory that firefox.exe exists in), in a traditional shell, the user has to add that directory to the Path variable of their shell. However, as explained before, this behavior isn't exactly easy to do, especially for a new user. With WELSH, only one command is needed, and the user does not have to manipulate the path variable.

WELSH also provides advanced Windows users another native option to their choices for a command-line shell on Windows. WELSH was created with the .NET framework, which interfaces with the Windows API to work on the wide range of Windows operating systems.

## **PROCESS**

This project originally started in March 2009 as a research project for an undergraduate Computer Systems course. The shell is programmed in C# using Visual Studio C# 2008 Express Edition. A preliminary version was released in May 2009, and another stable version mostly containing bug fixes of the initial version was released in July 2009.

Since then, the preliminary version is being developed as a honors project. New features in WELSH include a new and faster parser that uses theoretical parser methodologies, the ability to use loops and if/then statements, the ability to configure the shell, and remote access. Also, integrated testing has been included using NUnit, which is an integrated testing framework for the .NET languages. Integration tests were developed to make sure the parser works properly and is used for all support functions.

## **CAPABILITIES**

WELSH is capable of file management and navigation through the ability to copy, move, rename, and delete files or folders, as well as the ability to easily move through the file system using keywords such as "back," "forward" and "up."

WELSH is capable of process management, as well. A user can easily bring up a list of running processes and view each process's start time and how much virtual and working memory each process is using. The user can also kill processes easily either by the name of the process, or the process id.

WELSH features aliases for all built-in commands (see Figure 1) and allows the user to define new aliases for commands. The user can also create aliases for files, so that you can simply type "web browser" to access your web browser (see Figure 5). You can open non-executable files as well through aliases. The user can access web pages and local file sources directly in the command line. When opening files through an alias or file path, the resources

open in their default programs in Windows. For more information regarding this feature, see Figure 3 and Figure 4.

WELSH also features a unique security feature that allows a user to protect a file or folder. IN this way, the user can prevent the accidental deletion of files if a folder containing one of the said files is deleted while using WELSH. For example, if the user has a folder that was protected called "Swizzle" in a folder called "Sticks," WELSH will not allow the user to delete the folder "Sticks" because "Swizzle" is a protected file. By default, WELSH comes with the "Program Files" and "Windows" folders protected. Users can also toggle on and off whether or not a file is protected.

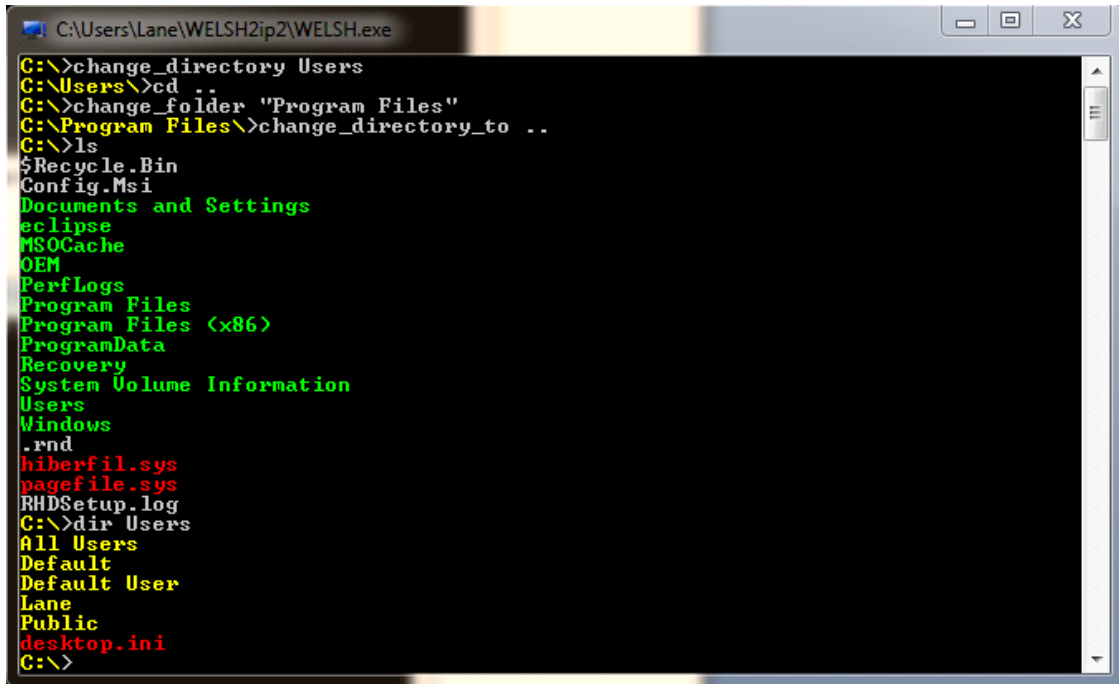
Piping is also allowed, but with one caveat - while Unix-style piping is evaluated from left to right, WELSH piping is evaluated right to left. For instance, the command *echo "!" | echo "?"* will produce the output of "?!" not "!?".

WELSH allows the user to create variables. Variables can be of two types: numbers (which are rational numbers) and strings. Each variable has a tag before its name to determine its type (@ for strings, # for numbers). Users can perform simple inline calculations involving addition (+), subtraction (-), multiplication (\*), division (/) and modulus (%) that use these variables, along with literals and other expressions. This makes it possible to use WELSH as a simple calculator, instead of having to open up a separate program.

WELSH has the capability for scripting and the ability to make scripts using all of the commands in WELSH. WELSH scripts can also have subroutines and the ability for unconditional jumps through the script. WELSH can evaluate Boolean-based (while statement) and counter-based loops as well (loop statement), and Boolean-based selection statements (if statement) (e.g. Figure 2).

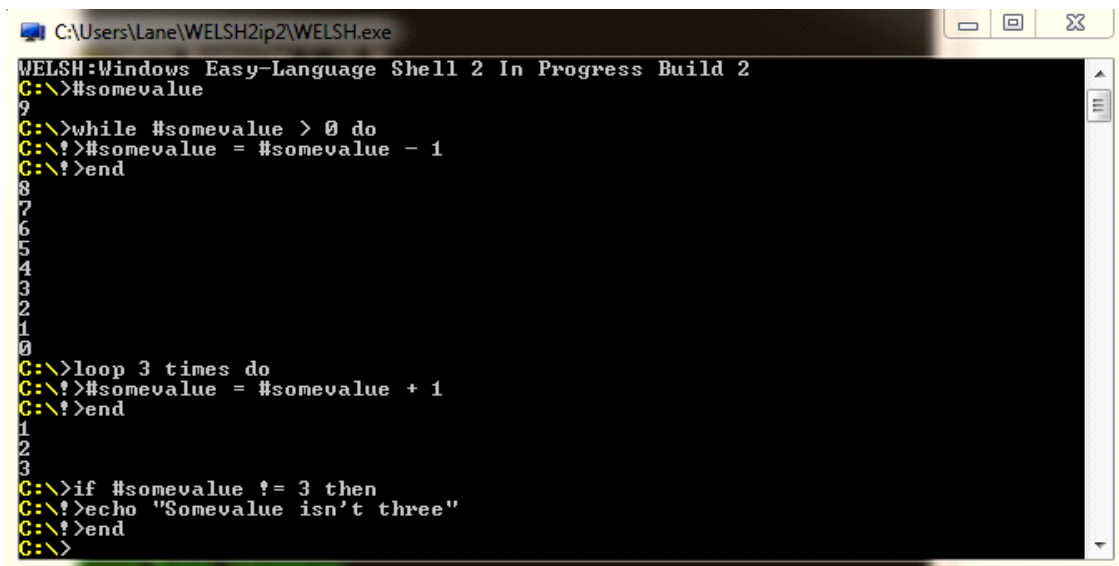
WELSH is easy to translate into different languages and easy to configure. There is a built-in configuration tool that allows the user to change the default foreground and background colors, allows the user to have password protection, and much more. The files that contain each string and command that WELSH uses are external user-readable and modifiable text files.

## FIGURES



```
C:\Users\Lane\WELSH2ip2\WELSH.exe
C:\>change_directory Users
C:\Users>cd ..
C:\>change_folder "Program Files"
C:\Program Files>change_directory_to ..
C:\>ls
$Recycle.Bin
Config.Msi
Documents and Settings
eclipse
MSOCache
OEM
PerfLogs
Program Files
Program Files (x86)
ProgramData
Recovery
System Volume Information
Users
Windows
.rnd
hiberfil.sys
pagefile.sys
RHDSetup.log
C:\>dir Users
All Users
Default
Default User
Lane
Public
desktop.ini
C:\>
```

Figure 1: This figure shows that the ls and cd commands can easily be activated using different commands, such as "change\_folder" and "dir."



```
C:\Users\Lane\WELSH2ip2\WELSH.exe
WELSH:Windows Easy-Language Shell 2 In Progress Build 2
C:\>#somevalue
9
C:\>while #somevalue > 0 do
C:\!>#somevalue = #somevalue - 1
C:\!>end
8
7
6
5
4
3
2
1
0
C:\>loop 3 times do
C:\!>#somevalue = #somevalue + 1
C:\!>end
1
2
3
C:\>if #somevalue != 3 then
C:\!>echo "Somevalue isn't three"
C:\!>end
C:\>
```

Figure 2: This figure shows that WELSH can interpret logic and iterative statements.

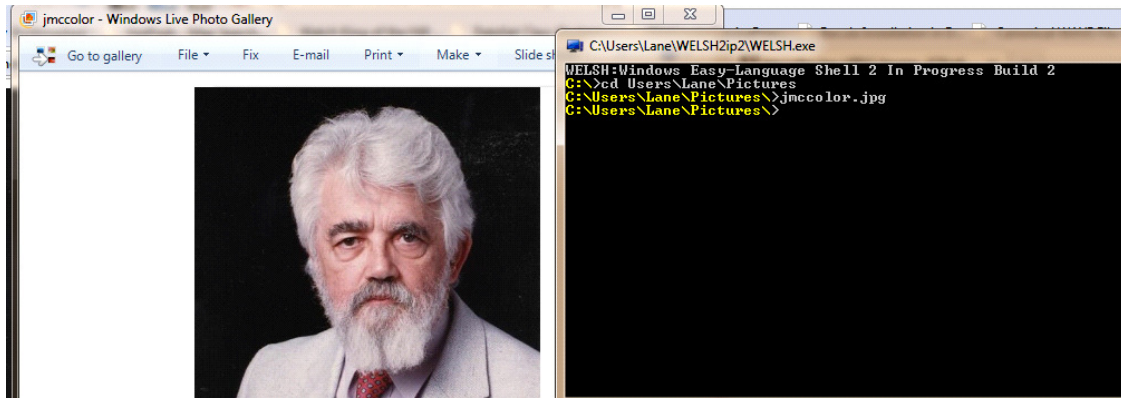


Figure 3: This figure shows that WELSH can open up files by just typing the file name. The file opens in whatever program is set up as the default program for that file type.

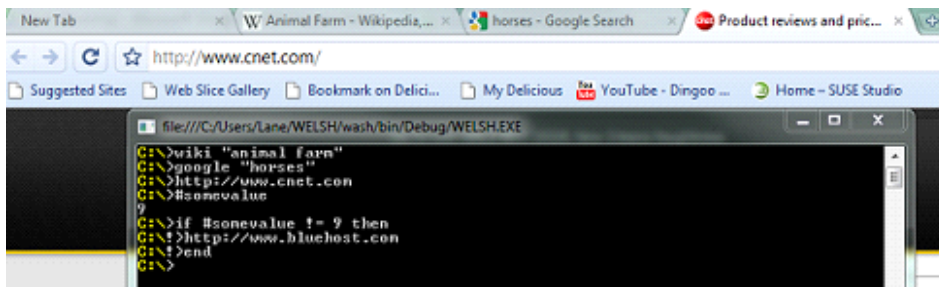


Figure 4: This figure shows that WELSH can open web pages, and has built-in commands for Google and Wikipedia.

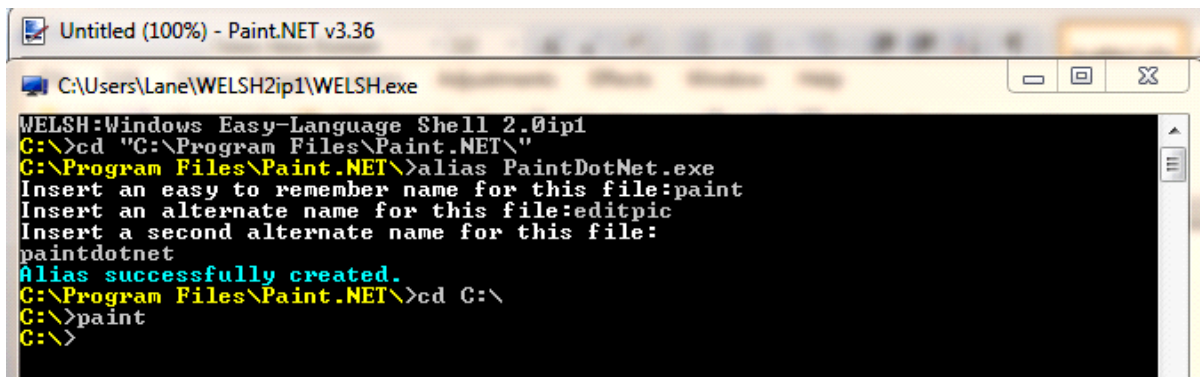


Figure 5: This figure shows how users create new aliases for files. Now, the user can simply type in "paint," "editpic" or "paintdotnet" to open Paint.NET.

## CONCLUSION

WELSH was designed with the average computer user in mind, that is, someone who types up a few emails a day, works on a few Word documents and surfs the Internet. It was not designed for the person who knows everything about computers and is willing to memorize

commands, but it is powerful enough for advanced users. As a result, WELSH uses commands that are typically easy to remember and recall and are similar to other commands used in other shell environments. Hopefully, this will attract the "average user" so that he or she may be willing to learn more advanced shells in the future.

WELSH is an ambitious and large software project and used knowledge from computer science disciplines such as operating systems, automata theory, compilation processes, client-server interactions, human-computer interaction and software engineering. So far, the project has developed well and is a ready-to-use application, and it continues to evolve as more features are added. The newest version of WELSH can be found at <http://www.1995again.com/windows-easy-language-shell/>.

## REFERENCES

1. Collyer, S. Shell Functions and Path Variables, part 1, March 2000, <http://www.linuxjournal.com/article/3645>, November 2009.
2. Edwards, E.D.N, The rise of the Graphical User Interface, December 1995, <http://people.rit.edu/easi/itd/itdv02n4/article3.htm>, November 2009.
3. Kernighan, B.W., *Unix For Beginners - Second Edition*, Murray Hill, NJ: Bell Laboratories, 1978.
4. Microsoft, Microsoft Windows XP - Command shell overview, <http://www.microsoft.com/resources/documentation/windows/xp/all/proddocs/en-us/cmd.mspx?mfr=true>, November 2009.
5. Payette, B. *Windows PowerShell in Action*: Manning Publications, 2007.
6. Robert, C.M., Myers, B.A., Integrating a Command Shell into a Web Browser, *Proceedings of the 2000 USENIX Annual Technical Conference*, 171-182, 2000.