A Tiny Windows PowerShell Guide

The intend of this guide is to provide a quick and painless introduction of Windows PowerShell and its tools, especially for users familiar with Unix and Linux shell tools.

Motivation

If you need shell tools for Windows, stop crippling yourself with old and unsuitable tools like Windows Command Prompt and Linux shell emulators. Seriously.

For power users, command prompt is simply not good enough. Yes, you can use Unix-style shell emulators as well, but they are hacky at the best. Granted that both the prompt and shell emulators can be used to do many of the everyday tasks, but the more complex your tasks get, the less these tools provide you with good facilities.

Windows is not Unix. Factually, these are two completely different ecosystems. Many of the design principles are shared, but the community and the tools are fundamentally different. Unix tools are made from the premises of the Unix world; applicability to Windows is only a potential afterthought. Similarly, it is a bad idea to assume Windows tools work smoothly in Linux.

If you want to work in Windows, I really encourage you to familiarize yourself with PowerShell. Yes, it requires an initial investment of time and effort. Risk it and you may find yourself becoming very comfortable and even more productive.

Quick Start

Open and Configure PowerShell

Find *Windows PowerShell* from the menu. Run it. If you are in Windows 7 or later, I strongly suggest pin it to your taskbar (Right click -> *Pin to taskbar*).

Personally I hate the default configuration. The bluish background makes my eyes bleed. The standard window size is too small. Especially as paths in Windows are typically very long, width of 80 is insufficient. Also QuickEdit should be on by default, but sadly, it isn't.

My preferred settings:

- **Settings**: check on *QuickEdit*
- **Font**: set type to `Consolas` and size to `14`
- **Window**: set buffer and window width `120` and window height to `40`

- **Colors**: set background to black (or whatever you prefer)

Restart and type `$Host.Version` to check which version of PowerShell you are running. You should end up seeing something like:

We are set to go!

Hints:

- When typing commands, use TAB for auto-completion
- Use `.\myprogram` to run a program or script (just like `./myprogram` in bash)

Cheatsheet

Get yourself started by using the following list of rough equivalents for typical Linux shell commands in PowerShell.

Bash	PowerShell (compatibility)	PowerShell (native)
`cd ../path`	`cd ..\path`	`set-location ..\path`
`ls`	`ls`	`get-childitem`
	`dir`	`gci`
`cat out.txt | more`	`cat out.txt | more`	`get-content out.txt | more`
		`gc file.txt | more`
`man command`	`man command`	`get-help command`
		`help command`
`mkdir dir`	`mkdir dir`	`mkdir dir`
		`md`
`rm -Rf dir`	`rm -recurse -force dir`	`remove-item -recurse -force dir`
		`ri -recurse -force dirname`
`cd ~`	`cd ~`	`set-location $env:home`
		`sl $env:home`
`ps`	`ps`	`get-process`
		`gps`
`ps -aux | grep proc`	`ps | grep proc`	`get-process proc*`
		`gps proc*`
`kill 1234`	`kill 1234`	`stop-process 1234`
		`spps 1234`
`touch file`[a]	n/a	`new-item file -type file`
		`ni file -type file`
`alias c=command`	n/a	`set-alias c=command`
		`sal c=command`

[a] Given that *file* doesn't exist.

Anatomy of PowerShell Commands

For compatibility with command prompt and Unix/Linux style shell usage, PowerShell uses aliases to its native commands. The native commands itself are often longer and often in the format of `Verb-Noun` such as `Set-Location`. For convenience, you do not need to obey letter casing. In fact, I prefer just writing commands in lowercase and auto-completing the name with *TAB* key if necessary. Aliases for native commands are often provided as well: for instance, `Set-Location` has a short-hand of `sl`.

The conceptual difference here is that commands in PowerShell are based on *cmdlets*. **A cmdlet** is a lightweight command that is used in the Windows PowerShell environment. By convention, cmdlets are named in the format of *Verb-Noun*. Examples include `Get-Help`, `Stop-Process` and `Format-List`. It is recommended that verbs are chosen from a list of approved alternatives (just write `Get-Verb` to see this list).

From the outside, cmdlets seem to work just like programs in Unix/Linux: they take text (parameters) as input and provide text as output within the processing pipeline. Externally cmdlets seem to be performing just that as well and can actually be used in this fashion "just as plain text" for instance with `grep`.

Instead of raw text, cmdlets operate on objects; they process input objects from the pipeline and typically deliver objects as an output to the pipeline. As such, instead of raw text separated by newline characters, cmdlets process *objects*, one at a time. Also instead of stand-alone executables, cmdlets are objects (instances of .NET Framework classes).

This all may sound heavy and scary, but it isn't that: cmdlets are basically just as easy to use as standard executables. In addition, cmdlets provide us with some advantanges. *Most notably you don't have to worry about input parsing, error presentation or output formatting since PowerShell runtime already does this for you!*

More Recipes

Find Occurrences of Given String from Files

Bash:

```
find . | xargs grep 'string' -ls
```

PowerShell (native):

```
gci | select-string "string"
```

Show Processes Consuming the Most CPU (`top`)

```
while (1) { cls; ps | sort -desc cpu | select -first 25; sleep 1}
```

List Environment Variables

```
gci env:
```

List USB Devices (`lsusb`)

```
gwmi Win32_USBControllerDevice
```

Download a File (`wget` or `curl`)

```
(new-object System.Net.WebClient).DownloadFile('http://example.com', 'output.html')
```

Why `curl http://example.com > output.html` may be a bad idea?

Default encoding in PowerShell is UTF-16, which may end up mangling your file.

Run process with elevated access (`sudo`)

@TBD@

Change File Permissions (`chmod`)

@TBD@

Change File Ownership (`chown`)

@TBD@

Rip CD/DVD (@TBD@)

Bash:

```
cat /dev/sdb > file.iso
or
dd if=/dev/sdb of=file.iso
```

Stuff: http://winserverteam.org.uk/blogs/austin/archive/2007/11/02/burn-cd-s-and-dvds-with-powershell.aspx

&&-pattern

Example:

```
mkdir test && cd test
```

TODO

Manage Swap (@TBD@)

Bash:

```
swapoff
swapon [file]
```

References and Further Reading

- [1] Cmdlet Overview (http://msdn.microsoft.com/en-us/library/windows/desktop/ms714395%28v=vs.85%29.aspx)
- Why isn't QuickEdit on by default in console windows?. (http://blogs.msdn.com/b/oldnewthing/archive/2007/09/13/4886108.aspx)
- PowerShell Community Extensions. Homepage. (https://pscx.codeplex.com/)
- Mastering PowerShell. Available also as an eBook. (http://powershell.com/cs/blogs/ebook/).
- PowerShell vs Bash Compared (PowerShell for Unixers) (http://xahlee.org/powershell/PowerShell_for_unixer.html)
- PowerShell as cmd.exe or Bash (http://xahlee.org/powershell/commands.html)
- An excellent article about curl, wget and PowerShell. (http://blog.commandlinekungfu.com/2009/11/episode-70-tangled-web.html)
- Bash vs PowerShell. Windows PowerShell Blog. (https://blogs.msdn.com/b/powershell/archive/2008/07/08/bash-vs-powershell.aspx)
- Writing Command-line Tools with IronPython and Visual Studio. PyCon 2011 Presentation. (http://www.slideshare.net/noahgift/iron-python-command-line)
- You can't be a 21st century admin without PowerShell. (http://blogs.technet.com/b/jamesone/archive/2009/11/02/you-can-t-be-a-21st-century-admin-without-po
aspx)
- PowerShell for Unix Hackers - Part I (https://docs.google.com/View?id=ap5pdc5jjps_58fs9cvwcz)

Copyright Information