C:/Users/troin/Dropbox/JH/core/IT_stack/onGitHub/OS-MSWin10/QR/cli

==========

```
chcp  # reports code page
~\AppData\Roaming\Vifm\vifmrc
```

in Explorer: `Alt+D > cmd > Enter`

Command Prompt

`PS> cmd` then `exit`

environment variables

```
echo %username%
echo %userprofile%
    echo %appdata%
    echo %localappdata%
set  # shows them all
```

computer name

```
echo %computername%
hostname
```

path

```
echo %path:;=&echo.%  # %PATH% on separate lines
powershell -Command ($env:Path).split(';')
pwsh -Command ($env:Path).split(';')
```

The Platinum Searcher

```
pt /version
pt search .
```

in $Drpbx\JH

```
pt '$core' .
pt '$DJH' .
pt '$JH' .
```

PowerShell

```
$MyInvocation.MyCommand.Name  # = the script's name
$MSwin10\gac.ps1  # to explore all commands
'"Hello world"'
1..3
<command> | Out-Null  # works for some commands
Alt > Space > E > L > [ up / down  to scroll   Esc ]
iex <someCommand>  # = invoke-expression
less <someFile>
foreach($element in 1..3){ $element }
get-package | Format-Table -AutoSize
PowerShell -NoProfile
sleep 1
```

$_ = $PSItem = current object in the pipeline object `measure-object` = `measure`

aliases

limited to single commands

Get-Alias

```
gal -Definition Get-ChildItem
gal ls
gal | sort Source | ft -view Source
```

standard

```
gal h*
nal <alias> <string>
sal <alias> <string>
```

cmdlets

```
get-commandSyntax <command>
```

- format-table: ft
- get-content: cat, gc, type
- get-item: gi
- invoke-expression: iex
- remove-item: del, erase, rd, ri, rm, rmdir

colour

```
[System.Enum]::GetValues('ConsoleColor') | ForEach-Object { Write-Host $_ -ForegroundColor $
iex "$ITstack\MSWin\PowerShell\colours\ConsoleColor.ps1"
iex "$ITstack\MSWin\PowerShell\colours\LindbergColors.ps1"
iex "$onGH\misc\Colors.ps1"
Write-Color -Text 'Red ', 'Green ', 'Yellow ' -Color Red,Green,Yellow
```

PSScriptTools

```
Get-PSSessionInfo
```

bars

```
New-ANSIBar -Range (232..255)
New-RedGreenGradient
```

data

```
$x.GetType()
Get-MyVariable
```

arrays

```
$a = 1,'a',2,'b'
foreach ($element in $myArray) {$element}
Format-Custom -InputObject $array -Expand CoreOnly  # displays structure
```

strings

```
"Hello".Replace('l', 'x').Replace('H', 'Y')
'a'.equals('b')
'me'+'et'
$string.trim()  # removes whitespaces (including newlines) from ends
[string]$Pwd
Format-String PowerShell -Randomize
if ( '5' -ne '4' ) { '5 is not 4' }
```

datetime

```
[System.TimeZoneinfo]::GetSystemTimeZones() | Out-GridView
Get-TimeZone -listavailable | Out-GridView
Get-TZData Australia/Hobart
```

Get-Date

```
$n = Get-Date -f yyyyMMddhhmmss
(Get-Date).Day
(Get-Date).ToString("yyMMdd-HHmmss")
```

no standard aliases

executables

```
cd 'C:\Program Files'
(Get-ItemProperty HKLM:\Software\Microsoft\Windows\CurrentVersion\Uninstall\* ).displayname
C:\MozillaThunderbird\thunderbird.exe -addressbook
C:\Windows\explorer.exe "microsoft-edge:searchterm"
gcm explorer
explorer shell:Appsfolder  # Applications
start <somefile>
where.exe gpg
where.exe pwsh
where.exe where.exe  # doesn't find executables in  ~\AppData\
```

foreach-object

- % = `foreach`
- not the `foreach` loop statement

help

```
Get-Help Start-BitsTransfer
man <cmdlet>  # Get-Help, paged
powershell /?
Update-Help -UICulture en-US
```

history

```
Get-LastModifiedFile
```

command history

```
gvim (Get-PSReadlineOption).HistorySavePath
h  # Get-History
```

networking

```
wp
```

IP

```
cc
ip
Get-WhoIs 8.8.8.8
```

modules

```
$env:psmodulepath -split (';')
C:\Users\troin\Documents\PowerShell\Modules
get-installedmodule
get-module
get-module -all
get-module -listavailable  # details, including old and those in  Windows PowerShell
gvim $env:localappdata\Roaming\Microsoft\Windows\PowerShell\PSReadLine\ConsoleHost_history.t
New-BurntToastNotification
pc  # PowerColorLS
```

PSScriptTools

```
Get-MyAlias  # limited to  PSScriptTools
Get-DirectoryInfo  # alias  dw
Get-MyVariable
Get-PathVariable
Get-PSSessionInfo
Get-PSScriptTools  # synopsi of commands
Get-TZData Europe/Paris
Get-WhoIs 8.8.8.8
Open-PSScriptToolsHelp
```

compact lists

```
Get-ModuleCommand ps.checkModuleUpdates
Get-ModuleCommand PSScriptTools
```

Show-Tree

```
pstree -?
t [n]  => pstree <depth>
```

`-InColor` only works in `Windows PowerShell`

PSFzf

```
Alt+a   # select an argument
Alt+c   # change to sub-directory
Ctrl+r  # search PSReadline History
Ctrl+t  # select provider path
Invoke-Fzf -?
```

tab completion

update

```
$machine\updateModules.ps1  # for reference
compare-module | where updateneeded | foreach { update-module $_.name }  # slow but reliable
```

Ruby

```
where.exe irb
where.exe ruby
```

scripts

```
C:\Users\troin\Documents\PowerShell\Scripts
Get-InstalledScript
param( [switch]$doSomething )  # -doSomething  creates  $doSomething = true
```

storage

```
dw  # directory counts
get-volume  # reports partitions
ii .  # invoke Explorer on WD
sl <directoryToMoveTo>
takeown /? | less
```

investigations

```
bat <textFile>  # with beautiful formatting and less paging
csl  # symlink targets in green or red
dw -?
f  # fzf preview files with bat
Get-FileHash <fileForWhichYouWantSHA256>
if ( ! ( test-path 'dir' ) ) { ni -name 'dir' -type directory }
pc -a -l -t  # includes hidden & time sorted
stringInVims "string"
```

Get-Content

```
gc <file>
gc somefile.txt | where { $_ -match  “expression” }
```

aliases: `cat`, `type`

by extension

```
cex
Get-FileExtensionInfo -r
tomalak
```

gci

```
dir, list, ls
gci * | select FullName
gci * | select Name
gci . -force  # Gets both hidden & non-hidden. Shows desired symlinks target.
gci -r -i i1*,i2*,i3*
gci -r -i '* (* conflicted copy *' |%{echo $_.fullname} | ri
gci -r | ? Name -match <regex>
```

PSAnsiMap
```
gci . | ft -view ansi  # overrides Terminal-Icons
Get-PSAnsiFileMap  # shows ANSI mappings
```
pure string output
```
gci *.pdf | select -expand name
gci $path | select -expand FullName
```
lastwritetime
```
lwp \.ps1
lwt md md
lwt sh sh
lwt docs doc odt
lwt tex cls sty tex
lwt txt txt
```
quick terminal output
```
lwts *.ps1 *.sh
lwts *.txt
```
sizes
```
du64 -l 1
Get-GitSize  # when you're in .git's parent directory
```
functions in $MSWin10\PSProfile.ps1
```
dc
fso
gfsi
```
manipulations
```
gci -r -i "*.txt" | %{mi $_.fullname ($_.fullname -replace ".txt",'.dw')}  # renames all txt
robocopy /mir <sourcedir> <destinationdir> /l  # runs a simulation of mirroring source to de
```
ROBOCOPY.exe
aliases
copy-item: copy cp cpi move-item: move, mv, mi remove-item: erase, del, rd, ri, rm, rmdir
rename-item: ren, rni
new-item
```
ni <directory> -type directory
```
`md` = `mkdir`, which calls `new-item`
system info
```
$Profile
(Get-CimInstance Win32_OperatingSystem) | Select-Object -Property Version, Caption
Get-PowerShellEngine -Detail
Get-PSLocation
Get-PSWho
Get-WindowsUpdateLog  # creates  \Users\troin\Desktop\WindowsUpdate.log
Sysdm.cpl -> System Properties
winver  # pops up  About Windows
Get-PSSessionInfo
Get-WindowsVersion
Get-WindowsVersionString
Test-IsElevated
```

environment variables

```
$env:computername
$env:localappdata
$env:pathext
$env:programfiles  # C:\Program Files
${env:programfiles(x86)}  # C:\Program Files (x86)
$env:username
$env:userprofile
$home
$psscriptroot
gci env:  # list the environment variables
```

Roaming

```
$Env:AppData  # C:\Users\...\AppData\Roaming
ls "$Env:AppData\Pandoc\defaults"
```

path

```
$env:Path -split ';'
Get-PathVariable
```

text wrangling

```
"Hello".Replace('l', 'x')
& "${Env:ProgramFiles(x86)}\Vim\vim82\gvim.exe" <textfile>
(Get-Content $file) -replace $regex, $newtext | Set-Content $file
```

Emacs

```
C:\Emacs\emacs-28.1\bin\emacs.exe -nw
C:\Users\troin\AppData\Roaming\.emacs.d
sl C:\Users\troin\AppData\Roaming\.emacs.d
~\AppData\Roaming\.spacemacs
```

version

```
$PSVersionTable
Get-ItemPropertyValue -Path "HKLM:\SOFTWARE\Microsoft\PowerShellCore\InstalledVersions\*" -N
```

robocopy

```
/l   - list only (= simulate)
/tee - output to console as well as log file
```

Windows PowerShell - modules

```
C:\Users\troin\Documents\WindowsPowerShell\Modules
get-installedmodule
get-module -listavailable  # details, including old
```

wt

- in Explorer: `Alt+D > wt > Enter`
- `win+;` emojis
- `win+r > wt` opens `wt` on `~`
- Windows Terminal

Settings

`Startup > Launch on machine startup`

- `alt+shift+-` = `split down`
- `Ctrl+,` = `Settings`

key-binds

`Settings > Actions`

- `alt+D` duplicate pane right
- `alt+_` split pane below
- `alt++` split pane right
- `alt+arrow` move focus
- `ctrl+c` copy text
- `ctrl+T` new tab
- `ctrl+W` close pane
- `ctrl+Shift+PgUp` scroll up a page
- `ctrl(+Shift)+Tab` move to next (previous) tab