Krystal Script
A Modern Alternative
To Shell Scripting

About Krystal Script

Krystal Script is a modern alternative to Bash scripting on linux and powershell for windows. Krystal Script is not a replacement for either of the shells, but as an easier alternative for beginners. A lot of beginners are scared to use the terminal. Krystal Script aims to solve that issue. In Krystal Script, you write code in plain english, and the Krystal Transpiler magically generates respective script files for the target Operating system. It produces a shell script on linux(`.sh`) and a powershell script on windows(`.ps1`) It makes development easier, because you can write code once-in Krystal Script and target both powershell and bash at the same time.

Krystal Script Cheat Sheet

```
-> Com: This is a comment // A line starting with Com: is a comment.
Single line Comments do not get copied over into the Generated script file.

-> ~Com: This is a comment // A line starting with ~Com: is also a comment.
Single line Comments starting with ~Com:are copied over to the generated
script file.

-> MCom
Things written between the MCom phrase become multi-line comments.
All multiline comments do get copied over to the generated script file.
-> Mcom

-> The Concept of tilda scope:
Krystal Script aims to provide direct intercompatibility with regular shell
  and powershell commands. In order to achieve this, we created something
```

called the **tilda scope**. Take a look at this picture:

```
~linux
echo "This is a regular shell command"
~linux
```

```
~win
New-Item -ItemType Directory "folder1"
~win
```

The code inside ~linux regular shell script. When a shell script is generated, all code inside the ~linux gets included in the shell script. But not in the powershell script. Similarly everything inside ~win gets included in the powershell script generated but not in the shell script. So if there ever comes a time when Krystal Script can't meet your needs and you have to use regular shell/powershell commands, you can do so without hesitation by wrapping them around the ~linux(for shell script) and ~win(for powershell script) scopes respectively.

-> Display("Prints to the console") //Prints to the console

-> Display(no new line "Does not add a new line at the end")//Prints to the console without adding a new line at the end

-> Display(literal "Literally displays the string !@#$%^&*()") // Prints whatever is in between the quotes. (Can be used to print special characters. For example $HOME on linux always evaluates to the home directory path, to print the literal string $HOME, you might need to use the literal keyword at the beginning)

-> Display(literal no new line "displays !#$%^&*() literally without a new line") // A combination of literal and no new line

->MakeFile("example.txt") // Creates a new file if one does not exist. Else it will update the timestamp of the file

->MakeFile(force "example.txt") // If a file with the name example exists.txt, it deletes the old one and creates a new empty file with the same name

->MakeFolder("example") // Creates a new folder, if one doesn't already exist

->MakeFolder(enable sub dirs "one/two/three/four") // Creates the directory tree one/two/three/four

->MakeFolder(force "example") // If a folder with the name example exists, it deletes the old one and creates a new empty folder with the same name