SECTION I. BASIC INFORMATIONS

What is FreakC ?

FreakC is a transpiled esoteric scripting programming language that... has nothing to do with the C programming language. This programming language's syntax is dirty and funny, but it can actually make many decent programs. The language is very similar to Batch, and it is also transpiled to Batch.

Why FreakC ?

If you are drunk or freaking crazy, FreakC is like Gordon Ramsay's Beef Wellington to homeless people. (Yes)

What is in the devkit ?

- A FreakC compiler
- Some optional utilities

Compiler's usage

In the "FreakC" folder, open cmd and type this command to compile and run the code:

```
freakc file_name
```

Example:

```
freakc "Examples/HelloWorld.fclang"
```

If you want to compile the code only, type:

```
freakc file_name --compile
```

If you want to compile the code and show the compiled code, type:

```
freakc file_name --candr
```

To show the current version of the devkit, type:

```
freakc --version
```

To create a new FreakC project quickly, you can type:

```
freakc project_name --create
```

The compiler only works on Windows, and can only be compiled to Batch.

In what part does FreakC upgrade from Batch ?

Well, FreakC is just an esoteric programming language, so you shouldn't expect FreakC to be upgraded from Batch, but here are some of them:

- Real while, do-while loops
- Additional variables like 'numpoop', 'numpuke', etc
- Additional functions like 'OddOrEven[fnc]', 'CoolHackerMan[fnc]',...
- Additional operators like "++" or "--"
- *mEMe*

Tutorials

READ SECTION II

Samples

https://github.com/nguyenphuminh/FreakC/tree/master/Examples

Convert FreakC to .EXE files

In "FreakC/Utilities/Scripts" there is a file called battoexe.bat which helps to convert .bat files to .exe files.

So to convert FreakC to .EXE files, you need to compile FreakC codes to Batch, and then convert the Batch file generated by simply dragging that Batch file onto battoexe.bat.

You can actually find plenty of other tools online that helps you to converts Batch files to EXE files.

FreakC in many IDEs

READ SECTION III

Frameworks

There are many frameworks in the utilities folder that you can use, I will definitely update the usage of them soon.

Note: I don't own those frameworks.

3.x Releases

3.x releases are the finest releases of FreakC, they removes all the unnecessary commands/functions, as well as adding many new useful features like:

- Useful variables
- Useful functionalities like if statements, for loops, while loop, do-while loop, "++" and "--" operators and more
- The compiler is much more well-built than all the previous releases
- ***mEMe***

Developers

- Owner/Developer: Nguyen Phu Minh

Setup FreakC's development environment

https://youtu.be/l_3sFSArQWg

Social medias

- FreakC's official announcement on youtube: https://youtu.be/0Pbah29aI4s
- FreakC on itch.io: https://npmgames.itch.io/freakc
- FreakC's offical page on Facebook: https://www.facebook.com/FreakC-Programming-Language-111425377421861
- FreakC on: https://esolangs.org

Languages based on FreakC

- VNC: https://github.com/nguyenphuminh/VNC

Copyrights and License

SECTION II. TUTORIALS

Disclaimer

This tutorial is not for beginners at programming because it is very difficult to understand.

Print text with Swear[fnc]

To print out a string or text, you use:

```
Swear[fnc] string
```

or you can prints text using:

```
SwearLine[fnc] string
```

The differences between Swear[fnc] and SwearLine[fnc] is that Swear[fnc] prints out normal text, if there are no text printed out, there might be some errors. But SwearLine[fnc] prints out a space then a text, if nothing is printed out, it will add a new line.

Store Data with Swear[fnc]

You can actually store data to a file by using:

```
Swear[fnc] data>>file
```

Special characters

Unlike any characters, "!" requires "^", for example:

```
::This would causes no errors
Swear[fnc] @#/

::This would cause error ("!" will not be shown)
Swear[fnc] !

::Correct codes:
Swear[fnc] ^!
```

The reason for this is that FreakC has: SETLOCAL ENABLEDELAYEDEXPANSION which causes "!" to not be used properly.

Unicode characters

To start using Unicode character, type:

```
FuckUnicode[fnc] >nul
```

So now you can use Unicode characters:

```
FuckUnicode[fnc] >nul
Swear[fnc] ă ê ©
:: Would print out "ă ê ©"
```

Prints out command's text

If you type:

```
Swear[fnc] Swear[fnc]
```

It won't print out "Swear[fnc]", but it will print out "echo", because FreakC will compiles any statement/command without caring if it's a used data or not. So to actually print out "Swear[fnc]", you will need to add "^" in any place of the command. Example:

```
Swear[fnc] S^wear[fnc]
:: This will print out "Swear[fnc]"
```

First note

To pause in FreakC, please type in:

```
GotoToilet[fnc]
```

You will see this command underneath, but you should you this command to pause your program.

Variables and data types

Data types

There are three data types in FreakC: integer, string.

Variables

To declare a variable, you use:

```
PoopString[fnc] variable_name=string_value
```

To do math equations, you use:

```
PoopInt[fnc] variable_name=integer_value
```

To declare a variable as an array, you use:

```
data_type arr[array_index]=
```

Ex:

```
PoopString[fnc] arr[0]=Hello
PoopInt[fnc] arr[1]=100
```

Note: There is no real Array in FreakC, the reason for naming arr[index] is to run for loops easier for all those "fake elements of a fake Array".

To declare a variable from user's input, you use:

```
PoopInput[fnc] variable_name=
```

Note: If you do this, it will prints out "Enter name:" right next to the input

```
PoopInput[fnc] variable_name=Enter name:
```

Maths

You can do Math equations with FreakC like this:

```
PoopInt[fnc] result=1+1+2+4
```

If you do this, variable "result" will be "Hel + lo"

```
PoopString[fnc] result=Hel + lo
```

Math operators

- () - grouping
- ! ~ - - unary operators
- * / % - arithmetic operators
- + - - arithmetic operators
- << >> - logical shift
- & - bitwise and
- ^ - bitwise exclusive or
- | - bitwise or
- = *= /= %= += -= - assignment
- &= ^= |= <<= >>= , - expression separator
- ++ -- - plus/minus 1

Some Math commands

Make a number squared

```
SquareMyBois[fnc] variable_without_%%
```

Make a number cubed

```
CubeMyBois[fnc] variable_without_%%
```

Print out an equation:

```
FuckUp[fnc] equation
```

Assign the result of an equation to %fhsolved%

```
FuckUpLess[fnc] equation
```

Merge strings

To merge strings, do this:

```
PoopString[fnc] str1=Hello
PoopString[fnc] str2=World
PoopString[fnc] str=%str1% %str2%
```

So the value of %str% is "Hello World"

Don't merge string like this:

```
PoopString[fnc] str1=Hello
PoopString[fnc] str2=World
PoopString[fnc] str=%str1% + %str2%
```

This time, the value will be "Hello + World"

Use variables in different commands

You can use variables in FreakC commands as %variable_name%

For example, to print out a variable, you can do this:

```
PoopString[fnc] result=Hello World^!
Swear[fnc] %result%
```

But, to print out an element of an array, you use:

```
PoopString[fnc] a[0]=Hello
Swear[fnc] %a[0]%
```

To print out every element of an array, you can write something like this:

```
OpenHouse[fnc] ENABLEDELAYEDEXPANSION
PoopInt[fnc] a[0]=100
PoopInt[fnc] a[1]=35
PoopInt[fnc] a[2]=20
LoopStuffs[fnc] %%n in (0,1,2) do (
    Swear[fnc] ^!a[%%n]^!
)
```

Actually, there is a whole another way to create an array:

```
PoopString[fnc] arr=1 2 3 4 5
ScanLetters[fnc] %%i in (%arr%) do (
    ::Print out every elements
    Swear[fnc] %%i
)
```

Local and global variables

You can declare a global variable by just using all the ways mentioned recently.

To declare a variable locally, you will need to use:

```
OpenHouse[fnc]
PoopInt[fnc] ans=100
CloseHouse[fnc]
```

OpenHouse[fnc] and CloseHouse[fnc] helps create a local environment.

In Batch, to use variable in for loops, or enables command prosessor's extensions, you would need:

```
setlocal ENABLEDELAYEDEXPANSION
setlocal ENABLEEXTENSIONS
```

You can also do that with OpenHouse[fnc]

```
OpenHouse[fnc] ENABLEDELAYEDEXPANSION
OpenHouse[fnc] ENABLEEXTENSIONS
```

Note: DELAYEDEXPANSION is already enabled in FreakC. So you wouldn't need to enable it actually.

Special variables

- %numpiss% - A variable with the value as random numbers from 1 to 9
- %numpuke% - A variable with the value as random numbers from 1 to 99
- %numpoop% - A variable with the value as random numbers from 1 to 999
- %numdiarrhea% - A variable with the value as random numbers from 1 to 9999
- %numbutt% - A variable with the value as random numbers from 1 to 99999
- %time% - A variable with the value as the current timer
- %date% - A variable with the value as the current date
- %cd% - A variable with the value as the current directory
- %errorlevel% - A variable with the value as the current Batch errorlevel value
- %cmdextversion% - A variable with the value as the current Command Processor Extensions version number
- %cmdcmdline% - A variable with the value as the original command line that invoked the Command Processor
- %path% - A variable with the value as all environment variables
- %highestnumanodenumber% - A variable with the value as the highest NUMA node number on this machine

There are a lot of special variables left, but you might not find uses for them

Implementations on variables

- %variable_name:~0,-2% - would extract all but the last 2 characters of the variable
- %variable_name:~-5% - would extract the last 5 characters of the variable
- %variable_name:~5% - would remove the first 5 characters of the variable
- %variable_name:str1=str2% - would replace str1 with str2
- %PATH:~10,5% - would expand the PATH environment variable, and then use only the 5 characters that begin at the 11th (offset 10) character of the expanded result. If the length is not specified, then it defaults to the remainder of the variable value. If either number (offset or length) is negative, then the number used is the length of the environment variable value added to the offset or length specified/li>

Another way to declare variables

- Shit[typ] - Declare a variable as a string
- Puke[typ] - Declare a variable as an integer
- Cum[typ] - Create a function/label
- Piss[typ] - Declare a variable from user's input

Notes

There are variables that you CAN NOT USE like: %a%, %printString%, %fccompile%, %fccompilename%, %fcread%.

Also, spaces in FreakC is extremely important, so if you declare a variable like this:

```
PoopInt[fnc] abc = 100
```

It will declare the "abc " variable, so if you prints it out like this, it will not work

```
Swear[fnc] %abc%
```

You will have to code like this:

```
Swear[fnc] %abc %
```

Then, it will prints out " 100"

Other thing that you should notice is that

```
PoopString[fnc] text
```

would return any variable begins with "text"

Comments

Single-line comment:

```
:: Comment
```

Another way:

```
rem comment
```

Multi-line comment:

```
[cmt] comment
[ecmt]
```

To write a comment that won't show up in the compiled codes, use:

```
[hcmt] comment
```

Labels and Goto statement

Labels helps you to jump to a state or pass parameters to execute tasks (somewhat procedural programming).

To create a label/procedure, you use:

```
PoopFnc[fnc] label_name
```

To jump to a label, you use:

```
EatFnc[fnc] label_name
```

To call a label/procedure, you use:

```
GrabFnc[fnc] :label_name
```

Or:

```
LickFnc[fnc] label_name
```

GrabFnc[fnc] can also targets file, while LickFnc[fnc] can only targets label/function. For example, you can execute files like this:

```
GrabFnc[fnc] file_name
```

Differences between EatFnc[fnc] and GrabFnc[fnc]

EatFnc[fnc] jumps to a function and will not execute the previous code while GrabFnc[fnc] use code from the function but still execute the previous code.

Also, GrabFnc[fnc] also supports parameters, which helps you a lot of time.

For examples, this code will print the sum of two parameters:

```
GrabFnc[fnc] :plus 1 2
PoopFnc[fnc] plus
PoopInt[fnc] ans=%~1 + %~2
Swear[fnc] %ans%
```

Note: Parameters in FreakC are %~1, %~2,...

Notes

To restart the program, you can type:

```
EatFnc[fnc] FreakCCompiled
```

It is because the compiled code of FreakC is in a main label/procedure "FreakCCompiled"

This code would still work eventhough it contains special character

```
PoopFnc[fnc] dsasd$ 123213 323
EatFnc[fnc] dsasd$ 123213 323
```

Object Oriented Programming

There is no such thing as "OOP on FreakC", but you can fake one, like this:

```
 ::Call the function "Car" with parameters to declares elements
GrabFnc[fnc] :Car "supercar" "20" "special"

::Would print out the element "gas" of "supercar"
Swear[fnc] %supercar.gas%

::Would create a function called "Car", then pass all the parameters to the new variables.
PoopFnc[fnc] Car
DoIfFalse[fnc] "%~1" == "" DoIfFalse[fnc] "%~2" == "" DoIfFalse[fnc] "%~3" == "" (
    PoopString[fnc] %~1.gas=%~2
    PoopString[fnc] %~1.model=%~3
)
Die[fnc] /b 0
```

If statements

To use if statement, type:

```
DoIf[fnc] condition command_to_execute
```

Example:

```
 PoopInt[fnc] abc=100
DoIf[fnc] %abc% == 100 (
    Swear[fnc] abc is equal to 100
)
```

You can actually use a Batch command in DoIf[fnc]. Example:

```
 PoopInt[fnc] abc=20
DoIf[fnc] %abc% == 20 (
    echo abc is equal to 100
)
```

All the comparison operators:

- "==" - Equal
- "EQU" - Equal
- "NEQ" - Not equal
- "LSS" - Less than
- "LEQ" - Less than or equal
- "GTE" - Greater than
- "GEQ" - Greater than or equal

Other kinds or if statements:

Execute if a file exists

```
DoIfExist[fnc] file_name (
    command_to_execute
)
```

Execute if a variable is defined

```
DoIfDefined[fnc] variable_name (
    command_to_execute
)
```

Execute if a condition is false

```
DoIfFalse[fnc] condtition (
    command_to_execute
)
```

Else

```
DoIf[fnc] condition (
    command_to_execute
) OrNot[fnc] (
    command_to_execute
)
```

Notes

To use if not for DoIfDefined[fnc] or DoIfExist[fnc], you can do this:

```
DoIfNotExist[fnc] file (
    command_to_execute
)
DoIfNotDefined[fnc] file (
    command_to_execute
)
```

Pressing keys with respond

To receive keys pressed, add:

```
TapSomeSht[fnc] key
```

For examples, if you want the users to press one in "wsad", type:

```
TapSomeSht[fnc] wsad
```

To perform any actions, you will need to use a special if statement:

```
DoIfTap[fnc] position_of_key_in_TapSomeSht[fnc]
```

Example:

```
TapSomeSht[fnc] wsad
DoIfTap[fnc] 4 (
    Swear[fnc] You pressed "D"
)
DoIfTap[fnc] 3 (
    Swear[fnc] You pressed "A"
)
DoIfTap[fnc] 2 (
    Swear[fnc] You pressed "S"
)
DoIfTap[fnc] 1 (
    Swear[fnc] You pressed "W"
)
```

If not for tapping

You can use:

```
DoIfNotTap[fnc]
```

to use if not.

Instant condition checking

You can type this to check if the condition is true or false instantly:

```
TryMeBtch[fnc] condition
```

Then, it will prints out "Yes" if true, "No" if false.

For example, this will prints out "Yes"

```
TryMeBtch[fnc] "hello" == "hello"
```

You can also do this:

```
TryMeHoe[fnc] condition
```

It will assign 'Yes' or 'No' to %tmhres%.

Loops

For loops

Loops from m to n

```
LoopStuffs[fnc] %%parameters in (start,step,end) do (
)
```

Loops through files rooted in a folder

```
ScanOrgans[fnc] drive/directory %%parameters in (file) do (
)
```

Loops through strings or strings in a file

```
ScanLetters[fnc] drive/directory %%parameters in (string/file) do (
)
```

Loops through a file

```
ScanDiaries[fnc] %%parameters in (set) do (
)
```

Loops through a folder

```
ScanDir[fnc] %%parameters in (folder) do (
)
```

While loops

While loops can be created using:

```
WhileSuck[fnc] condition
::code
EndSuck[fnc]
```

Of course, the loops will run when the condition is still true, stop when false.

Do-While loop

```
RepeatSuck[fnc]
::code
UntilStale[fnc] condition
```

Differences between While and Do-While

While loop is executed only when given condition is true. Whereas, do-while loop is executed for first time irrespective of the condition. After executing while loop for first time, then condition is checked.

Notes when using while loops

YOU CAN NOT USE NESTED WHILE LOOPS
This will be wrong:

```
WhileSuck[fnc] condition
WhileSuck[fnc] condition
EndSuck[fnc]
EndSuck[fnc]
```

(Same with Do-While)

The better way for loops

You can use somewhat "recursion" like this:

```
PoopInt[fnc] i=start_number
PoopFnc[fnc] loop
command
DoIf[fnc] %i% == end_number EatFnc[fnc] nextcode
PoopInt[fnc] i++
EatFnc[fnc] loop

PoopFnc[fnc] nextcode
command
```

For example, this program will print all the number from 0 to 10 then print out "Done!":

```
PoopInt[fnc] i=0
PoopFnc[fnc] loop
Swear[fnc] %i%
DoIf[fnc] %i% == 10 EatFnc[fnc] nextcode
PoopInt[fnc] i++
EatFnc[fnc] loop

PoopFnc[fnc] nextcode
Swear[fnc] Done^!
```

Use Batch in FreakC

Unlike all the previous release, now FreakC support writing Batch without any other commands. So you can actually learn Batch and write Batch codes in FreakC.

Is adding Batch to FreakC will kill the usages of FreakC ?

Well, I would say it will and won't at the same time. What I mean is that if there is Batch in FreakC, you would definitely write Batch rather than FreakC, but actually, FreakC codes are just compiled to Batch anyway. The only key feature of FreakC is being an esolang with funny and dirty syntax, that's why people would want to use it. FreakC doesn't improve Batch, it's just a funny version of it.

Nul in FreakC

It's just like nul in Batch, so if you want to make your console not print out any process, you can do it like this:

```
Command >nul
```

To hide errors, you can do this:

```
Command >nul 2>nul
```

Creating and Inserting modules

You can create a module file using:

```
Cancer[fnc]
```

Note: The "Cancer[fnc]" command needs to stay at the top of the file.

Then, include it in the main file using:

```
InsertShits[fnc] module_name.bat
::Note: You have to replace .fclang with .bat
```

Then, compile the module file first, then compile the main file at the end.

Example:

Create a file called "module.fclang" with:

```
Cancer[fnc]
Swear[fnc] World
```

Create a file called "program.fclang" with:

```
Swear[fnc] Hello
InsertShits[fnc] module.bat
```

In the command window, type:

```
freakc module --compile
freakc program
```

It will prints out:

```
 Hello
World
```

If you are a C++ dev, this would be an equivalent to "#include".

Better way

If you don't want to merge all the codes into one file (as FreakC uses a lot of labels), you can simply:

1. Create a module file with Cancer[fnc] on top of all the codes and add Die[fnc] /b in the bottom.
2. Create a main file, but rather than using InsertShits[fnc], you can use normal GrabFnc[fnc] to access a file as a procedure.
3. Compile the module file first, then compile and run the main file.

A demo can be found on https://github.com/nguyenphuminh/FreakC/tree/master/Examples/Module.

Find strings in a file

FindShits[fnc]:

```
FindShits[fnc] [/V] [/C] [/N] [/I] [/OFF[LINE]] "string" [[drive:][path]filename[ ...]]

/V         Displays all lines NOT containing the specified string.
/C         Displays only the count of lines containing the string.
/N         Displays line numbers with the displayed lines.
/I         Ignores the case of characters when searching for the string.
/OFF[LINE] Do not skip files with offline attribute set.
"string"   Specifies the text string to find.
[drive:][path]filename Specifies a file or files to search.

If a path is not specified, FIND searches the text typed at the prompt or piped from another command.
If a path is not specified, FIND searches the text typed at the prompt or piped from another command.
```

FindTrash[fnc]:

```
FindTrash[fnc] [/B] [/E] [/L] [/R] [/S] [/I] [/X] [/V] [/N] [/M] [/O] [/P] [/F:file]
    [/C:string] [/G:file] [/D:dir list] [/A:color attributes] [/OFF[LINE]]
    strings [[drive:][path]filename[ ...]]

/B         Matches pattern if at the beginning of a line.
/E         Matches pattern if at the end of a line.
/L         Uses search strings literally.
/R         Uses search strings as regular expressions.
/S         Searches for matching files in the current directory and all subdirectories.
/I         Specifies that the search is not to be case-sensitive.
/X         Prints lines that match exactly.
/V         Prints only lines that do not contain a match.
/N         Prints the line number before each line that matches.
/M         Prints only the filename if a file contains a match.
/O         Prints character offset before each matching line.
/P         Skip files with non-printable characters.
/OFF[LINE] Do not skip files with offline attribute set.
/A:attr    Specifies color attribute with two hex digits. See "color /?"
/F:file    Reads file list from the specified file(/ stands for console).
/C:string  Uses specified string as a literal search string.
/G:file    Gets search strings from the specified file(/ stands for console).
/D:dir     Search a semicolon delimited list of directories
strings    Text to be searched for.
[drive:][path]filename Specifies a file or files to search.

Use spaces to separate multiple search strings unless the argument is prefixed with /C.
For example, 'FINDSTR "hello there" x.y' searches for "hello" or "there" in file x.y.
'FINDSTR /C:"hello there" x.y' searches for "hello there" in file x.y.

Regular expression quick reference:
.        Wildcard: any character
*        Repeat: zero or more occurrences of previous character or class
^        Line position: beginning of line
$        Line position: end of line
[class]  Character class: any one character in set
[^class] Inverse class: any one character not in set
[x-y]    Range: any characters within the specified range
\x       Escape: literal use of metacharacter x
\<xyz    Word position: beginning of word
xyz\>    Word position: end of word
```

(Copied from the documentation of Batch)

Other useful commands

Exit the program

```
Die[fnc]
```

Shutdown system

```
ShutdownSystem[fnc]
::Add /t time and /c "comment" to set the time to shutdown and leave a comment before shutdown
```

Restart system

```
RestartSystem[fnc]
::Add /t time and /c "comment" to set the time to shutdown and leave a comment before restart
```

Create a folder

```
PukeDir[fnc] folder_name
```

Access a folder

```
EatDir[fnc] folder_name
```

Access a drive

```
EatDrive[fnc] drive_name
```

Read a file

```
StealDiary[fnc] file_name
```

Clear the screen

```
Forget[fnc]
```

Pause

```
GotoToilet[fnc]
```

Delete a file

```
Trash[fnc]
```

Delete a folder

```
TrashDir[fnc] folder_name
```

Check if a number is odd or is even

```
OddOrEven[fnc] number
```

Change color, a pair of hex code is a color code

```
ChangeColor[fnc] hex_code
```

- 0 = Black
- 1 = Blue
- 2 = Green
- 3 = Aqua
- 4 = Red
- 5 = Purple
- 6 = Yellow
- 7 = White
- 8 = Gray
- 9 = Light Blue
- A = Light Green
- B = Light Aqua
- C = Light Red
- D = Light Purple
- E = Light Yellow
- F = Bright White

Change the title of the program

```
ChangeName[fnc] title_name
```

Change console's size

```
ChangeSize[fnc] size_number
ChangeSize[fnc] con cols=columns_or_width lines=lines_or_height
```

Rename a file

```
RenameFile[fnc] file_name
```

Move a file to the new path

```
MoveFile[fnc] file_name new_path
```

Copy a file to the new path

```
CopyFile[fnc] file_name new_path
```

Open a file or a website url (would open cmd if nothing is opened)

```
PlayFile[fnc] file
```

Timeout for a specific time

```
WaitForBus[fnc] time_as_second
```

Restart the program or loop the program endlessly

```
Loop[fnc]
```

Shows date

```
SeeDate[fnc]
```

Shows time

```
SeeTime[fnc]
```

Shows all files in the current directory

```
MyMemory[fnc]
```

Show command prompt content

```
EnableCoolStuff[fnc]
```

This would be useful for debugging.

Hide command prompt content

```
DisableCoolStuff[fnc]
```

Note: All the command prompt content is hiden by default

Prompt for date to change date

```
BuyCalender[fnc]
```

Note: Administrator is required to run the command

Prompt for time to change time

```
BuyClock[fnc]
```

Open powershell

```
DumbCousin[fnc]
```

Note: Administrator is required to run the command

Disable prebuilt variables

You can disable numpoop, numpiss, numpuke,... by putting this block of code on top of your program:

```
Cancer[fnc]
EnableCoolStuff[fnc]
```

Meme and J4F commands

Patrick functions

You can prints out a quote of patrick from the huge collection of... 3 quotes that FreakC comes with :))

```
 ::Is mayonnaise an instrument ?
Patrick[meme] 1
::Roses are blue, violets are red, I have to go to the bathroom.
Patrick[meme] 2
::I can't see my forehead.
Patrick[meme] 3
::The inner machinations of my mind are an enigma.
Patrick[meme] 4
::Once upon a time there was an ugly barnacle. He was so ugly that everyone died. The end!
Patrick[meme] 5
::I wumbo, you wumbo, he she we wumbo.
Patrick[meme] 6
::Two words, SpongeBob. Na. Chos.
Patrick[meme] 7
```

And yes, no more [fnc], there is [meme] now =D

Dani functions

You can prints out Dani's quotes :))

```
 ::Not as thicc as your mom.
Dani[meme] mom
::Prints "Wow that was really cool." endlessly
Dani[meme] cool
::Freak you billy.
Dani[meme] billy
```

SpongeBob functions

You can also do SpongeBob stuffs

```
::echo Ravioli, ravioli, give me formuoli.
SpongeBob[meme] 1
```

Others

Collections of stupid quotes

```
::Prints out Hello, World
HelloWorld[str]
::Prints out Grab me a drink, mate
GrabMeADrink[str]
::Prints out I Love You
ILoveYou[str]
::Prints out Are ya winning son ?
AreYaWinningSon[str]
::Prints out Hi mate
HiM8[str]
::Prints out Damn boi... Damn boi... Damn boi he thicc boia, that's a thicc ass boi.
Thicc[meme]
```

And again, yes, there is also [str] just for varieties =D

Prints out random numbers continously (like in the Matrix)

```
CoolHackerMan[fnc]
```

Java and JSON supports :))

Just for surprising elements, just try it on your own =D

```
Java[fnc]
Java[str]
Java[meme]
Json[str]
```

Equivalent with Batch

- Swear[fnc] - echo
- SwearLine[fnc] - echo.
- PoopInt[fnc] - set /a
- PoopString[fnc] - set
- PoopInput[fnc] - set /p
- PoopFnc[fnc] - :label
- GrabFnc[fnc] - goto
- Shit[typ] - set
- Puke[typ] - set /a
- Cum[typ] - :label
- Piss[typ] - set /p
- LoopStuffs[fnc] - for /l
- ScanOrgans[fnc] - for /r
- ScanLetters[fnc] - for /f
- ScanDiaries[fnc] - for
- ScanDir[fnc] - for /d
- DoIf[fnc] - if
- DoIfExist[fnc] - if exist
- DoIfDefined[fnc] - if defined
- DoIfFalse[fnc] - if not
- DoIfNotExist[fnc] - if not exist
- DoIfNotDefined[fnc] - if not defined
- DoIfTap[fnc] - if errorlevel
- DoIfNotTap[fnc] - if not errorlevel

- TapSomeSht[fnc] - choice /c: /n
- OrNot[fnc] - else
- GotoToilet[fnc] - pause
- Die[fnc] - exit
- ShutdownSystem[fnc] - shutdown -s
- RestartSystem[fnc] - shutdown -r
- PukeDir[fnc] - md
- EatDir[fnc]
- EatDrive[fnc] - drive:
- StealDiary[fnc] - type
- GrabFnc[fnc] - call
- Forget[fnc] - cls
- Trash[fnc] - del
- TrashDir[fnc] - rmdir
- ChangeColor[fnc] - color
- ChangeName[fnc] - title
- ChangeSize[fnc] - mode
- RenameFile[fnc] - ren
- MoveFile[fnc] - move
- CopyFile[fnc] - copy
- PlayFile[fnc] - start
- WaitForBus[fnc] - timeout
- SeeDate[fnc] - date /t
- SeeTime[fnc] - time /t
- MyMemory[fnc] - dir
- EnableCoolStuff[fnc] - @echo off
- DisableCoolStuff[fnc] - @echo on
- BuyCalender[fnc] - date
- BuyClock[fnc] - time
- OpenHouse[fnc] - setlocal
- CloseHouse[fnc] - endlocal
- FindTrash[fnc] - findstr
- FindShits[fnc] - find

Notes

Please do not use two or more commands with these commands:

- TapSomeSht[fnc]
- TryMeBtch[fnc]
- EatDrive[fnc]
- OddOrEven[fnc]

Also, FreakC is case-sensitive, so you need to be strict with your code.

Most of the commands are just modified Batch commands, so you can actually apply Batch logic in it.

SECTION III. IDE

VSCode

Download the "FreakC" extension from Marketplace and you are ready to go! The extension provides syntax-highlighting as well as code auto-completion. You can find the source code of the extension at https://github.com/nguyenphuminh/FreakC-vscode

Sublime text 3

In "FreakC/Utilities/Scripts" there is a file called "FreakC.sublime-build" which is the Sublime Text's build system for FreakC. To use it, please paste it in the "C:\Users\admin\AppData\Roaming\Sublime Text 3\Packages\User" or wherever your Packages folder is. Then, make sure that you have set the environment variable for FreakC. After that, you will be able to compiles FreakC codes in Sublime Text.

For syntax highlighting, copy the "FreakC" folder in the same folder, then paste it in "%APPDATA%\Sublime Text v\Packages\".

SECTION IV. LICENSE