makeDoc

Purpose	:	Export Documentation from taged lines.
File	:	makeDoc.sh
Author	:	Leandro - leandrohuff@programmer.net
Date	:	2025-09-21
Version	:	1.0.0
Copyright	:	CC01 1.0 Universal

Note: Changes in this document will be discarded on next build, any changes should be made on source code documentation instead.

Details

Save formatted lines from source code into documentation file.
Read source code line-by-line and save prefixed lines by tag ??D to file.
C/C++ source code lines start with tag //D and Bash lines start with tag ##D.
Only those lines started by tags are exportedd to documentation files.
Mixed commented lines can co-exist at same source code, one for local documentation purpose and another to be exported to apropriate documentation file.
All lines are documented using Markdown format, the exported document can be read by an Markdown program reader.

Index

Top	
Details	
Glossary	
Constants	
Variables	
Functions	
logFail	Print a failure log message
unsetVars	Unset global variables
_exit	End log, stop libShell, deinitialize variables and exit
printHelp	Print an help message
parseArgs	Parse parameters from command line
barGraph	Draw a progressive line counter bar graph
saveHeaderTo	Save a pre formatted HTML Header
saveFooterTo	Save a pre formatted HTML Footer
libShell	Source libShell
runScript	Main shell script application
Start Script	Start Shell Script
Bottom	

Top | Index | Bottom

Glossary

Use	Description
Constants	Memory space for read only data
Variables	Memory space for read/write data
Functions	Source/Executable statement code, can be called anywhere from source code
Parameters	Data passed to functions
Result	Functions result after execution
Return	Allways an integer returned from function to inform success or failure
none	Is similar as a void type, no parameter, no result or no return from function
char	One byte data type to store single characters
string	Char vector to store a group of characters
integer	Memory space to store ordinal numbers
float	Memory space to store 32 bits floating point numbers
double	Memory space to store 64 bits floating point numbers
type[]	Memory vector space to store contigous data type
##D	Bash, Zsh, Python, Perl, Ruby
//D	C/C++, C#, Java, JavaScript, Pascal/Object Pascal, Go, Swift, Kotlin, Rust
--D	SQL, Ada, Haskell
"D	Visual Basic, VBScript
%%D	LaTex, MATLAB

Top | Index | Bottom

Constants

integer[] **numVERSION** = (*1 0 0*)
integer[] **dateVERSION** = (*2025 9 21*)

Top | Index | Bottom

Variables

string **Source** : Source file to generate documentation from.
string **Destine** : Destine file to save documentation into.
boolean **isHtmlFile** : Flag to store HTML file detected.

Top | Index | Bottom

Functions

logFail( )

none **logFail**(*string* **"$*"**) : *string*
Send formatted failure log messages to screen.

Parameter:
string: **"$*"** - Message to display on screen.

Result:
string: Log message.

Return:
none

Top | Index | Bottom

unsetVars( )

integer **unsetVars**(*none*) : *none*
Unset global variables.

Parameter:
none

Result:
none

Return:
integer: **0** - Success

Top | Index | Bottom

_exit( )

integer **_exit**(*integer* **$1**) : *none*
Finish script file and return an exit code.

- Log runtime message.
- Finish log messages.
- Stop libShell.
- Unset global variables.
- Exit an error code.

Parameter:
integer: **$1** - Exit code.

Result:
none

Return:
integer: **0** - Success
integer: **1..N** - Error code.

Top | Index | Bottom

printHelp( )

integer **printHelp**(*none*) : *string*
Print an help information.

Parameter:
none

Result:

string: Help message on screen.

Return:
integer: **0** - Success

Top | Index | Bottom

parseArgs( )

integer **parseArgs**(*string* "**$@**") : *none*
Parse all parameters from command line.

Parameter:
-h - Print help information about syntax and use.
[*file*] - Open file as input and save in a file with extension *.md

Options:
-i *file* - Generate documentation from input file.
-o *file* - Generate documentation into output file.
-- [*parameters*] - Send [parameters] to libShell.

Result:
none

Return:
integer: **0** - Success
integer: **1..N** - Error code.

Top | Index | Bottom

barGraph( )

none **barGraph**(*integer* **counter**) : *string*
Draw a prograssive line counter bar graph.

Parameter:
integer : **counter** - Progress counter.

Result:
string : Draw a progressive counter bar graph accorgin to lines read for file.

Return:
none

Top | Index | Bottom

saveHeaderTo( )

integer **saveHeaderTo**(*string* **title** , *string* **file**) : *string*
Save a pre formatted HTML Header into a target file passed by parameter.

Parameter:
string : **title** - HTML title, if empty, file name will be used instead.
string : **file** - Target filename.

Result:
string : Pre formatted HTML header to save into target file.

Return:
integer : **0** - Success
integer : **1** - Error code, empty parameter or file not found.

Top | Index | Bottom

saveFooterTo( )

integer **saveFooterTo**(*string* **file**) : *string*
Save a pre defined HTML Footer into file.

Parameter:
string : **file** - Target file.

Result:
string : Pre formatted HTML Footer to save into a target file passed by parameter.

Return:
integer : **0** - Success
integer : **1** - Error code, empty parameter or file not found.

Top | Index | Bottom

Source and Initialize libShell

source **libShell.sh**
libInit
libSetup *-v -l 1*
logBegin

Top | Index | Bottom

runScript( )

integer **runScript**(*string* "**$@**") : *none*
Run bash script file.

Parameter:
string: "**$@**" - All command line parameters.

Result:
none

Return:
integer: **0** - Success
integer: **1..N** - Error code.

Top | Index | Bottom

Start Shell Script

runScript "*$@*"
Call function runScript( ) and pass all parameters from command line.

Top | Index | Bottom