makeDoc

```
            Export Documentation from taged lines.
     File : makeDoc.sh
     Date : 2025-10-05
   Author : Leandro
  Version : 2.1.1 / 2025-9-30
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

Glossary

```
        Use  Description
 Constants : Memory space for read only data.
 Variables : Memory space for read/write data.
 Functions : fileSOURCE/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.
```

Tags Programming Language Documentation
```
##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.
??E>: End, stop documentation until ??B> tag.
??B>: Begin, start documentation until ??E> tag.
??M>: Macro, evaluate lines to show variables values.
```

[Top] | [Index] | [Bottom]

Constants

Type	Name	Value	Description
integer[]	numVERSION	= (2 1 1)	Version Number.
integer[]	dateVERSION	= (2025 9 30)	Date Version Number.
integer	iFILE	= 0	Configuration file ID.
integer	iUSER_DIR	= 1	ID for user directory.
integer	iBASE_DIR	= 2	ID for base directory.
integer	iAPP_DIR	= 3	ID for base application directory.
integer	iDOC_DIR	= 4	ID for base documentation directory.
integer	iICON_FAILURE	= 5	ID for Failure icon file.
integer	iICON_SUCCESS	= 6	ID for Success icon file.
integer	iICON_NOK	= 7	ID for Not Ok icon file.
integer	iICON_OK	= 8	ID for Ok icon file.
integer	iLOG_TARGET	= 9	ID for log target (screen|file).
integer	iLOG_LEVEL	= 10	ID for log level.
integer	iMAX	= 11	Number of IDs.
integer[]	tableID	= [1]	IDs table.
string[]	tableTAG	= [2]	Tags table.
string[]	tableDEFAULT	= [3]	Default values table.

[1] =(0 1 2 3 4 5 6 7 8 9 10)

[2] =(FILE USER_DIR BASE_DIR APP_DIR DOC_DIR ICON_FAILURE ICON_SUCCESS ICON_NOK ICON_OK LOG_TARGET LOG_LEVEL)

[3] =(makeDoc.cfg \$HOME dev makeDoc doc icons/failure.png icons/success.png icons/nok.png icons/ok.png 1 -v)

[Top] | [Index] | [Bottom]

Variables

Type	Name	Value	Description
boolean	enableDOC	= true	Enable documentation.
boolean	flagFORCE	= false	Assume yes for any question.
string	fileSOURCE	= ''	Source file to generate documentation from code.
string	fileDESTINE	= ''	Destine file to save documentation into.
integer	exitCODE	= 0	Store exit code from main program.
string	userMESSAGE	= ''	Formatted message for message box.
string	tableCONFIG	= ''	Table for user configuration values.
string	configFile	= ''	User configuration filename.
string	currentDir	= ''	Current directory.
string	userDIR	= ''	User home directory.
string	baseDIR	= ''	Base development directory.
string	appDIR	= ''	Application directory.
string	docDIR	= ''	Documentation directory.
string	iconFAIL	= ''	Icon failure directory.
string	iconSUCCESS	= ''	Icon success directory.
string	iconNOK	= ''	Icon not Ok directory.
string	iconOK	= ''	Icon Ok directory.

[Top] | [Index] | [Bottom]

Functions

logFail( )

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

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

Result:
 string: Log message.

Return:
 none

unsetVars( )

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

Parameter:
 none

Result:
 none

Return:
 integer: **0** – Success

_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.

printHelp( )

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

Parameter:
 none

Result:
 string: Help message on screen.

Return:
 integer: **0** – Success

getTag( )

none **getTag**(*string* **line**) : *string*
 Get tag name from a string until '=' symbol.

Parameter:
 string : **line** – Get the tag name from string line.

Result:
 string : Echoe the 'tag' name from line string.

Return:
 none

getValue( )

none **getValue**(*string* **line**) : *string*
 Get tag value from a string after '=' symbol.

Parameter:
 string : **line** – Get the tag value from string line.

Result:
 string : Echoe the 'tag' value from line string.

Return:
 none

saveUserConfigToFile( )

integer **saveUserConfigToFile**(*string* **filename**) : *string*
 Save a pre formatted configuration into a filename.

Parameter:
 string : **filename** – Filename to save user configuration content.

Result:
 string : Save a pre formatted configuration into a filename.

Return:
 integer : **0..N** – Return code.
 integer : **0** – Success
 integer : **1** – Empty parameter.
 integer : **2** – Could not save to file.

loadUserConfigFromFile( )

integer **loadUserConfigFromFile**(*string* **filename**) : *string*
 Load configuration list from a file or a default configuration table.

Parameter:
 string : **filename** – Filename to load user configuration content.

Result:
 string[] **tableCONFIG** : Store configuration from filename.

Return:
 integer : **0..N** – Return code.
 integer : **0** – Success
 integer : **1** – Empty parameter or file acces is not available.

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.

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]](#)

guiMessageBox( )

integer **guiMessageBox**(*string* **title** , *string* **text** , *string* **image**) : *none*
 Dialog box to show a message and get answer.

Parameter:
 string : **title** – Dialog box title.
 string : **text** – Dialog box text.
 string : **image** – Dialog box image.

Result:
 none
Return:
 integer : **0** – User choose Ok.
 integer : **1** – User choose Close.

[[Top]](#) | [[Index]](#) | [[Bottom]](#)

guiShowMessage( )

integer **guiShowMessage**(*string* **title** , *string* **text** , *string* **image**) : *none*
 Dialog box to show a message.

Parameter:
 string : **title** – Dialog box title.
 string : **text** – Dialog box text.
 string : **image** – Dialog box image.

Result:
 none
Return:
 integer : **0** – User choose Ok.
 integer : **1** – User choose Close.

[[Top]](#) | [[Index]](#) | [[Bottom]](#)

loadConfiguration( )

integer **loadConfiguration**(*string* **file**) : *none*
 Load user configuration values from file.

Parameter:
 string : **file** – Configuration filename.

Result:
 none
Return:
 integer : **0** – Success
 integer : **1..N** –Failure

[[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]

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]

Load and Initialize libShell

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

[[Top] | [[Index] | [[Bottom]

Start Shell Script

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

[[Top] | [[Index] | [[Bottom]