blib Code Reference v1.1

David Hobach

Contents

blib

blib - a bash library

The basic functions which are imported by default.

version: Execute `blib version` or use b_version.

Disclaimer

You should have received a copy of the Lesser GNU General Public License along with this program. If not, see https://www.gnu.org/licenses/.

The above statements apply to all modules of blib if not mentioned otherwise.

Coding Conventions

1. embrace the KISS principle
2. some general guidelines: http://www.kfirlavi.com/blog/2012/11/14/defensive-bash-programming/
3. use b_[module]_[camel case function name] to denote functions meant to be used by users of the library, B_[module]_[upper case var name] for global variables
4. use BLIB_[module]_[upper case var name] for global variables (to be avoided whenever possible) not meant to be used by library users ("private" variables); library users can use setters or getters
5. use the BLIB_STORE with the above naming conventions for "private" variables whenever possible
6. use blib_[module]_[camel case function name] to denote private functions not meant to be used by library users; the function name should not contain any underscores
7. the blib module itself is the only exception which can use b_, B_, blib_, BLIB_ without module name
8. prefixes such as t_, T_ and UTD_ are exclusively related to test code
9. set exit codes (!= 0 –> issue) wherever it makes sense
10. keep the global namespace clean whenever possible
11. declare -g must be used in order to allow sourcing from a function (b_import)
12. modules must implement a function such as
    b_[module]_getDeps
    returns: newline-separated list of dependencies of the module
13. modules should provide a header similar to that of the blib source file in order to make the documentation generation work
14. modules may be placed in subfolders of arbitrary depth
15. use 0 to indicate true and 1 to indicate false for variables; for exit codes use a non-zero exit code to indicate the number of errors
16. functions should be tagged with the following tags, if applicable:
    @StateChanging - the function changes the internal state of the script in a way that will not propagate to supershells (e.g. global variables) and should thus not be called from subshells (unless the user wants the state to only change in that shell)
    @B_E - the function uses B_E for error handling and may thus behave differently depending on the implemented error handler

Library Usage

with the default bash options:

```
source blib
b_import [module]
```

Dependencies

dirname
readlink
whoami
sort
cat
mktemp
rm
su

Global Variables

B_TEST_MODE

blib/B_TEST_MODE

blib will set this variable to 0, if blib is running in test mode.
This variable may be used to bypass code during testing, if bats cannot test that code due to its limitations (e.g. for EXIT traps which bats uses for itself).

B_CALLER_NAME

blib/B_CALLER_NAME

Name of the executable or script as String which called blib and any child libraries.

B_ERR

blib/B_ERR

Global variable used for error handling throughout blib, cf. B_E.

It is recommended to always set an at least partially static error message on confirmed errors as variables may be empty (which would indicate "no error" for B_E).

B_RC

blib/B_RC

Can be used to set the return code for B_E in the case of an error. It defaults to 1.

This should be set to an integer value between 1 and 255. Any other value may cause undefined behaviour.

B_LIB_DIR

blib/B_LIB_DIR

Path of the blib installation directory.

B_SCRIPT

blib/B_SCRIPT

Global variable which can be used to obtain the two global variables B_SCRIPT_DIR and B_SCRIPT_NAME as follows:

```
eval "$B_SCRIPT"
```

B_SCRIPT_NAME

blib/B_SCRIPT_NAME

Path of the sourced or executed bash script executing the eval (symlinks are resolved) of B_SCRIPT.

B_SCRIPT_DIR

blib/B_SCRIPT_DIR

Name of the sourced or executed bash script executing the eval (symlinks are resolved) of B_SCRIPT.

Global Aliases

Alias expansion is automatically enabled by blib as it is required for its core functionality. So if you have strange aliases defined in your shell environment, this may cause undefined blib behaviour.

B_E

blib/B_E

The blib error handler: All blib modules use it whenever execution errors require special handling that the currently executing code cannot achieve.

Syntax:

```
B_ERR="This is an error message." ; B_E ;
```

If you need to set the return/exit code, you can do it with B_RC:

```
B_ERR="This is another error message." ; B_RC=6 ; B_E ;
```

Calling B_E means:
Check B_ERR for an error message and if there is one, handle it. It can be placed at the end of a line or on its own line. B_E will then process the error message in the way defined by the error handler (cf. b_defaultErrorHandler) and stop any further execution of code in the current context (function, script, ...) returning a non-zero exit code (1) unless the described error was fixed. In the latter unlikely case it'll let execution proceed.

The error handler can be re-defined at runtime with b_setErrorHandler.

Functions

b_printStackTrace [skip level]

blib/b_printStackTrace

[skip level]: skip that many levels of the stack trace (optional, default: 1 - skip this function call)
print the current stack trace in a human readable way

returns: stack trace with the first levels skipped as defined

b_nop

blib/b_nop

Do nothing.

returns: nothing; sets a zero exit code

b_version [part]

blib/b_version

Get the version of this blib instance.

[part]: Optional parameter defining the part of the version to retrieve (0: all as String (default), 1: major as Integer, 2: minor as Integer).

returns: blib version as string; always sets a zero exit code

b_defaultErrorHandler [error out] [print stderr] [print stack trace]

blib/b_defaultErrorHandler

The blib default error handler.
As any error handler it must

1. handle the error message (if not the error itself) lying in B_ERR
2. not take any non-numeric arguments
3. not error out itself
4. implement the below *[error out]* as its first parameter (to make b_setBE work)
5. return one of the following exit codes:
   a) 0: **if and only if** the error was fixed entirely and the caller may ignore the error (i.e. probably never)
   b) 1: The error wasn't fixed. Functions should return to their caller indicating an error (non-zero status code). Direct shell calls will exit. B_ERR is **not** reset to blank, i.e. the next call to B_E in the same context will cause another error. The caller may use this to either *throw* the error further or handle and clear the error.
   c) 2: Force a stop of execution in the current shell / error out.

[error out]: Whether or not to call exit after the error message handling, if the error couldn't be handled (default: 0 = always error out / call exit). If set to 1, B_E will allow e.g. functions to return to their callers.

[print stderr]: Whether or not to print the error message to stderr (default: 0 = print).

[print stack trace]: Whether or not to print a stack trace to stderr (default: 0 = print).

returns: see the description above

b_setBE [error out]

blib/b_setBE

Set the [error out] behaviour of the currently configured error handler.

Contrary to b_setErrorHandler this function may be called by blib modules as all error handlers are required to support [error out] as parameter.

Example for switching the error out behaviour:

```
b_setBE 1
funcThatMayCallB_E #without subshell
ret=$?
b_resetErrorHandler
```

[error out]: see b_defaultErrorHandler (default: 0)

returns: nothing, always sets a zero exit code

@StateChanging

b_setErrorHandler [handler]

blib/b_setErrorHandler

Set the error handler for all future exections of B_E in the current scope.

You can do this in e.g. subshells to limit the effect.

blib modules should only use this function if absolutely necessary to temporarily modify the error behaviour whilst making sure that b_resetErrorHandler is called in the end. Otherwise it will prevent library users from setting the general behaviour in their scripts.

Usually you do not want to write an entirely new handler, but modify the b_defaultErrorHandler parameters with this setter or use b_setBE for that.

[handler]: Function to handle errors. See b_defaultErrorHandler for details.

returns: nothing

@StateChanging

b_resetErrorHandler

blib/b_resetErrorHandler

Set the error handler to whatever it was before the last call to b_setErrorHandler or b_setBE.

returns: nothing, always sets a zero exit code

@StateChanging

b_getErrorHandler

blib/b_getErrorHandler

Get the currently for B_E configured error handler.

returns: the error handler function

b_silence [function] [param 1] .. [param p]

blib/b_silence

Call the given function with its parameters in the current shell context whilst suppressing all of its output to both stdout and stderr. Anything written to B_ERR however is passed to B_E (which can still write to stderr).

This function is useful when you want to keep an error message set with B_ERR, but discard everything else.
In contrast `yourfunction &> /dev/null` may also drop the error message, if you're using an error handler (see B_E) that writes to stdout or stderr.

[function]: The function to execute.

[param p]: An arbitrary number of function parameters.

returns: Sets the status code of the called function, but doesn't print anything. B_E is called on errors.

@B_E

b_info [message]

blib/b_info

Print the given message to stdout as info message.

[message]: to print to stdout

returns: nothing

b_enforceUser [user name]

blib/b_enforceUser

enforce that the user is the given one and if not, exit the script and set a non-zero status code

[user name]: user name to check against

returns: nothing

@B_E

b_isFunction [potential function name]

blib/b_isFunction

check whether the given function is defined

returns: zero exit code if the function is defined

b_getBlibModules

blib/b_getBlibModules

get all available blib module names as a newline-separated list

returns: all available blib module names as newline-separated list

b_listContains [list] [entry]

blib/b_listContains

check whether the given list contains the given entry

[list]: newline-separated list

[entry]: string to be found on a single line within the list (equality check)

returns: a zero exit code if the list contains the entry; a non-zero exit code otherwise

b_checkDeps [list]

blib/b_checkDeps

check whether the given list of dependencies is met by the system running this function.

[list]: newline-separated list of binaries or commands that the system must be able to execute

returns: list of dependencies not met; a non-zero exit code is set, if the list contains elements

b_blib_getDeps

blib/b_blib_getDeps

Get the dependencies of this module.

returns: newline-separated list of dependencies of this module

b_import [module] [double import]

blib/b_import

Import the given module into the current context.

[module]: relative path of the module to import (relative to the blib/lib root directory)

[double import]: if set to 1, import the given module regardless of whether it was imported before (default: 0 = don't do duplicate imports)

returns: nothing, errors out if the import failed and sets a non-zero status code; if the import was successful or previously done, a zero exit code is set

@StateChanging

@B_E

b_generateStandalone [function] [param 1] .. [param p] - [module dep 1] .. [module dep n] - [function dep 1] .. [function dep d]

blib/b_generateStandalone

Create a standalone variant of blib in a single file running the given function when called (sourcing that file will only make the functions available) and print that file to stdout.

The current execution state is not retained.

[function]: The function to call when the generated script is executed. All script parameters when calling [output file] are passed to this function. The function must be available in the current context.

[param p]: Static parameters to add to the function as single String. Dynamic parameters should be passed to the generated script.

[module dep i]: Names of the modules to include in the standalone file. They do not need to be imported.

[-]: Dash used as separator between the function dependencies and the modules. If none is provided, all parameters are assumed to be modules.

[function dep j]: An arbitrary number of functions that need to be added in order to satisfy the dependencies of the function to call (e.g. if function A is meant to be called, but uses function B internally, you'll have to pass B as one of its dependencies). Dependencies that can be found in added modules should *not* be added.

returns: Sets a zero exit code and prints the output file to stdout on success. May error out otherwise.

@B_E

b_execFuncAs [user] [function] [param 1] .. [param p] - [module dep 1] .. [module dep n] - [function dep 1] .. [function dep d]

blib/b_execFuncAs

Attempt to execute the Bash function as the given user.

Whether or not this works highly depends on the underlying OS and its (sudo & su) configuration. In particular this function may cause further execution to wait for the user to type in the password of the requested user.

If the given user is identical to the current user, b_execFuncAs may decide to run the function in the current context. Otherwise it may run in a different process.

[user]: User to execute the function as (default: root).

[function]: The function to execute.

[param p]: An arbitrary number of function parameters.

[-]: A dash as separator character between the various parameters.

[module dep i]: Names of the modules required by the function. They do not need to be imported by the function itself.

[function dep j]: An arbitrary number of functions that need to be added in order to satisfy the dependencies of the function to call (e.g. if function A is meant to be called, but uses function B internally, you'll have to pass B as one of its dependencies). Dependencies that can be found in added modules should *not* be added.

returns: Whatever the executed function returns. A non-zero exit code may also indicate that the user switch didn't work. In particular B_E is *not* called if the executed function returns an error.

@B_E

b_isModule [module name]

blib/b_isModule

Test whether the given name represents a blib module name.

returns: sets a zero exit code if the given name is a valid module name

arr

Collection of array related functions.

Copyright (C) 2018 David Hobach LGPLv3
0.2

Dependencies

printf

Functions

b_arr_getDeps

arr/b_arr_getDeps

Get the dependencies of this module.

returns: newline-separated list of dependencies of this module

b_arr_join [delimiter] [array]

arr/b_arr_join

Join the given array; elements are separated with the given delimiter. The array is not checked to exist.

[delimiter]: String to use as delimiter.

[array]: expanded array to join, e.g. "${arr[@]}"

returns: Joined version of the array. The exit code is always zero.

b_arr_contains [element] [array]

arr/b_arr_contains

Check whether an array contains an element.

[element]: element to check for its existence in the array

[array]: expanded array to check, e.g. "${arr[@]}"

returns: an exit code of 0, if the element was found and 1 otherwise

b_arr_mapsAreEqual [map spec 1] [map spec 2]

arr/b_arr_mapsAreEqual

Check whether the two given maps/associative arrays are equal.

[map spec 1]: First map specification to check. Since maps cannot be passed directly to functions in Bash 4.2, you'll have to use `"$(declare -p "yourmap")"` instead.

[map spec 1]: Second map specification to check.

returns: an exit code of 0, if the maps are equal and 1 otherwise; B_E is only triggered on programming errors

@B_E

cdoc

Generate code documentation in many formats (e.g. html, pdf, manpage, ...) from code comments.

Lines applicable for the documentation in your code are assumed to match static (configurable) regular expressions. These lines are then fed to pandoc in order to generate a single html page (or pdf, manpage, ...) as documentation. If no conversion is required (input format = desired output format), pandoc is bypassed.

It should be possible to use this way of generating code documentation with most programming languages (incl. bash). The defaults however are set for bash and the blib way of documenting its code, i.e. you'll have to use the getters and setters of this module if you want something different. For instance the default is to check for lines starting with #+ (a special bash comment line) and add everything afterwards to the output documentation.

Various callback functions can be used to add content to the output of b_cdoc_generate. See the documentation of that function for details.

0.2

Dependencies

mv
rm
mktemp

Functions

b_cdoc_getDeps

cdoc/b_cdoc_getDeps

Get the dependencies of this module.

returns: newline-separated list of dependencies of this module

b_cdoc_setExtractionRegex [regex]

cdoc/b_cdoc_setExtractionRegex

Set the regular expression used to check for matching lines in code files. The first match (${BASH_REMATCH[1]}) is added to the documentation output.

returns: nothing

@StateChanging

b_cdoc_getExtractionRegex

cdoc/b_cdoc_getExtractionRegex

See the setter.

returns: The property that was set.

b_cdoc_setFileFilterCallback [callback function name]

cdoc/b_cdoc_setFileFilterCallback

Set the function to call by b_cdoc_generate exactly once right after it computed the list of applicable source code files.

The callback function should be declared as follows:

```
callback_function_name [file list]
[file list]: newline-separated list of files (no directories!)
             that b_cdoc_generate computed for document generation
             (in that order)
returns:     the newline-separated list to use by b_cdoc_generate for
             document generation (default: the input); a non-zero exit
             code will abort further processing
```

returns: nothing

@StateChanging

b_cdoc_getFileFilterCallback

cdoc/b_cdoc_getFileFilterCallback

See the setter.

returns: The property that was set.

b_cdoc_setDocumentBeginCallback [callback function name]

cdoc/b_cdoc_setDocumentBeginCallback

Set the function to call by b_cdoc_generate exactly once right before it starts generating the output document.

The callback function should be declared as follows:

```
callback_function_name [document output file] [document output format]
[document output file]: path to the document output file
          (may not exist and should not be written to)
```

```
[document output format]: chosen output format
returns:  whatever should be added at the beginning of the output document;
          a non-zero exit code will abort further processing
```

returns: nothing

@StateChanging

b_cdoc_getDocumentBeginCallback

cdoc/b_cdoc_getDocumentBeginCallback

See the setter.

returns: The property that was set.

b_cdoc_setPostProcessingCallback [callback function name]

cdoc/b_cdoc_setPostProcessingCallback

Set the function to call by b_cdoc_generate each time a code file was fully processed.

The callback function should be declared as follows:

```
callback_function_name [processed input] [input file] [document output format]
[processed input]: Everything that was found to match the
                   extraction regex in the [input file] by b_cdoc_generate.
[input file]:      The original input file.
[document output format]: chosen output format
returns:           whatever should be added to the output document for the
                   given input file (usually the processed input or some filtered
                   version of it); a non-zero exit code will abort further processing
```

returns: nothing

@StateChanging

b_cdoc_getPostProcessingCallback

cdoc/b_cdoc_getPostProcessingCallback

See the setter.

returns: The property that was set.

b_cdoc_setDocumentEndCallback [callback function name]

cdoc/b_cdoc_setDocumentEndCallback

Set the function to call by b_cdoc_generate exactly once right after it generated the output document.

The callback function should be declared as follows:

```
callback_function_name [document output file] [document output format]
[document output file]: path to the document output file
                        (may not exist and should not be written to)
[document output format]: chosen output format
returns:                whatever should be added to the end of the output
                        document; a non-zero exit code will abort further processing
```

returns: nothing

@StateChanging

b_cdoc_getDocumentEndCallback

cdoc/b_cdoc_getDocumentEndCallback

See the setter.

returns: The property that was set.

b_cdoc_setSpaceCallback [callback function name]

cdoc/b_cdoc_setSpaceCallback

Set the function to call by b_cdoc_generate each time it hits "space" (non-matching lines) between two matching lines.

The callback function should be declared as follows:

```
callback_function_name [matching line] [input file] [previous space count] [document output
[matching line]: The first matching line before which no match was found.
[input file]:    The original input file.
[previous space count]: The number of times this function was previously
                called for the currently processed file.
[document output format]: chosen output format
returns:        whatever should be added in front of the matching line;
                a non-zero exit code will abort further processing
```

returns: nothing

@StateChanging

b_cdoc_getSpaceCallback

cdoc/b_cdoc_getSpaceCallback

See the setter.

returns: The property that was set.

b_cdoc_generate [input files] [output file] [output format] [additional pandoc options]

cdoc/b_cdoc_generate

Generate a documentation file from the given list of input files or directories.

pseudo code description for the document generation:

1. call the file filter callback to obtain the final list of input source code files to use for document generation, respect the order (default: use all files passed as input)
2. call the document begin callback function with the output file path to get user-specific output (default: do nothing)
3. for all input files:
    i. for all lines of a file:
        a) store lines matching b_cdoc_getExtractionRegex in a variable *o*
        b) between any two matching lines for b_cdoc_getExtractionRegex that had a non-matching line in between them: call the space callback function (default: add an empty line to the output)
    ii. pass *o* and the file name to the post processing callback function (default: return the input) - users could add e.g. the file name as section header here
    iii. add the output of the post processing function to the output document
4. call the document end callback function with the output file path to get user-specific output (default: do nothing)
5. do all necessary output conversions using pandoc

[input files]: Newline-separated list of files or directories to generate the documentation from. The given order is respected; directories are recursively searched for files. It is currently assumed that these files are encoded in UTF-8.

[output file]: Path to the documentation file to generate. Should not exist.

[output format]: The target format of the documentation to generate. See pandoc for a list of available output formats. If none is specified, pandoc is bypassed and the input format is chosen as output format. Passing "pandoc" will let pandoc decide based on the extension of the output file.

[additional pandoc options]: All remainining parameters will be directly passed to pandoc. If none are provided, -s is implicitly added as default.

returns: Sets a non-zero exit code and exits the script on errors. output from pandoc and other calls may be printed. Otherwise nothing is returned.

@B_E

b_cdoc_generateBlibStyle [input files] [output file base path] [output format] [delete existing]

cdoc/b_cdoc_generateBlibStyle

A convenience wrapper for b_cdoc_generate which sets various reasonable parameters depending on the output format.

[input files]: see b_cdoc_generate

[output file base path]: path to a directory and base file name where to store the generated output documentation file; the final file name may differ as it is chosen by this function

[output format]: currently one of raw|html|pdf|man is supported (default: raw)

[delete existing]: whether or not to delete previously created output files (default: true/0); if set to false (1), the function will error out if a previously created file was found

returns: full path to the created documentation file on success; otherwise the function may error out

@B_E

Callback Functions

b_cdoc_cbPrintNewline

cdoc/b_cdoc_cbPrintNewline

Prints a newline character.

returns: nothing

b_cdoc_cbPrintFirstParam [param]

cdoc/b_cdoc_cbPrintFirstParam

Prints the first parameter.

[param]: The parameter to print.

returns: the first parameter

daemon

Module providing access to a single background process providing some service (daemon). Exiting the foreground control process will *not* terminate the background process. Attempting to start multiple background daemons will be prevented in a thread-safe way.

Each background process is assumed to implement a `daemon_main` function and must be identified by a unique String.

If you need to exchange data with the background service, please have a look at the multithreading/ipcv or multithreading/ipcm modules.

0.3

Dependencies

kill
umask
mkdir

Imports

multithreading/mtx

proc

Global Variables

B_DAEMON_ID

daemon/B_DAEMON_ID

Contains the ID of the daemon, if and only if the current process is the daemon process.

Functions

b_daemon_getDeps

daemon/b_daemon_getDeps

Get the dependencies of this module.

returns: newline-separated list of dependencies of this module

b_daemon_init [quiet flag] [main name] [stdout file] [stderr file] [umask setting]

daemon/b_daemon_init

Init the module paramters. It is necessary to call this method *before* using any other of this module unless you want to use the default paramters.

[quiet flag]: If set to 0 (default), don't print anything to stdout during the execution of start|stop|restart|status. Otherwise use b_info to print informational messages.

[main name]: Name of the main loop function to execute in the background process. Returning from that function will exit the background process. Default: `daemon_main`

[stdout file]: Where the background process should write its stdout stream to (default: /dev/null).

[stderr file]: Where the background process should write its stderr stream to (default: /dev/null).

[umask settings]: The umask settings to apply to the daemon (default: 0).

returns: Nothing.

@StateChanging

b_daemon_start [id] [arg 1] ... [arg n]

daemon/b_daemon_start

Start the background process.
If you need to start it as a different user, simply run this function as a different user with e.g. (b_execFuncAs)[#b_execFuncAs]. Please keep in mind that control processes must have the permission to send signals to the daemon PID though.

[id]: Unique identifying String of the daemon to distinguish it from others.

[arg i]: An arbitrary number of arguments which can be passed to the main loop.

returns: Sets a zero exit code on success. Otherwise sets a non-zero exit code. In particular B_E is called, if the daemon is already running.

@B_E

b_daemon_stop [id] [kill timeout]

daemon/b_daemon_stop

Stop the background process.

[id]: Unique identifying String of the daemon to distinguish it from others.

[kill timeout]: Time in seconds after which the background process will be killed, if it remains unresponsive (default: 2).

returns: Sets a zero exit code, if the daemon terminated by itself. An exit code of 2 indicates that the daemon had to be killed. An exit code of 3 means that it wasn't running. B_E is called on unexpected errors.

@B_E

b_daemon_restart [id] [kill timeout] [arg 1] ... [arg n]

daemon/b_daemon_restart

Restart the background process.

[id]: Unique identifying String of the daemon to distinguish it from others.

[kill timeout]: See b_daemon_stop.

[arg i]: An arbitrary number of arguments which can be passed to the main loop.

returns: See b_daemon_start.

@B_E

b_daemon_statusPid [id]

daemon/b_daemon_statusPid

Check the status of the background process.
Doesn't print informational messages to stdout.

[id]: Unique identifying String of the daemon to distinguish it from others.

returns: The PID and sets a zero exit code, if the daemon is running and a non-zero exit code otherwise. B_E is only called on exceptional errors.

@B_E

b_daemon_status [id]

daemon/b_daemon_status

Check the status of the background process and print informational messages to stdout (if configured).

[id]: Unique identifying String of the daemon to distinguish it from others.

returns: Sets a zero exit code, if the daemon is running and a non-zero exit code otherwise. B_E is only called on exceptional errors.

@B_E

b_daemon_getPid [id]

daemon/b_daemon_getPid

Get the process ID of the background process.

[id]: Unique identifying String of the daemon to distinguish it from others.

returns: The process ID and sets a zero exit code, if it could be obtained. Please note that the process may be dead anyway (use b_daemon_statusPid for that). Otherwise a non-zero exit code is set. B_E is only called on exceptional errors.

@B_E

date

Collection of date and time related functions.

0.2

Dependencies

date

Functions

b_date_getDeps

date/b_date_getDeps

Get the dependencies of this module.

returns: newline-separated list of dependencies of this module

b_date_addDays [date] [days] [format] [utc flag]

date/b_date_addDays

Add the given number of days to the given date.

[date]: date to add days to; the format must be understood by the Unix date utility

[days]: number of days to add

[format]: output format of the date, in Unix date notation (default: use the localized output)

[utc flag]: if set to 0, use UTC as time zone if not specified for the input *and* use it for the output (default: 1 = local time zone)

returns: The input date with the given number of days added, in the requested format; returns a non-zero exit code on errors.

@B_E

b_date_diffSeconds [date 1] [date 2]

date/b_date_diffSeconds

Get the number of seconds between the two dates, i.e. [date 2] - [date 1].

[date 2], [date 1]: the two dates to subtract; the time one is assumed to be identical if not specified within the dates

returns: The number of seconds between the given two dates [date 2] - [date 1]; returns a non-zero exit code on errors.

@B_E

flog

Flexible log writer for bash supporting arbitrary piped output (files, network streams, stdout, stderr, ...) in a user-defined output format.
Log files can be automatically reduced to their last X lines.

In order to log to the system log, please use the logger command instead. This library is mostly meant for application logs handled in a more custom manner.

Currently only a single writer per instance of this library/thread is kept in memory, but you can user multiple one after another or in multiple threads. Each writer should have a dedicated log file to write to.

Exact format of log entries:

```
[header][message]
[header]: Can be arbitrarily defined in the
          respective callback function. If nothing
          is defined, the below default header
          is used:
[default header] = '[default date] '
[default date]: current date in the format as used
          by date +"%F %T %Z" (the format can be changed)
```


0.3

Dependencies

date
tail
cat
mktemp

Imports

fs

Global Variables

B_FLOG_SEV

flog/B_FLOG_SEV

Global map for human readable severities which may be used by users of this script.
It was inspired by the severities of RFC5424.
Currently supported values: emergency|alert|critical|crit|error|err|warning|warn|notice|informational|info|debug

Functions

b_flog_printSeverity [severity]

flog/b_flog_printSeverity

[severity]: see b_flog_init
Print the given severity in a way for logging. This function is meant to be used as building block for header functions.

returns: a printed version of the given severity for logging

b_flog_close

flog/b_flog_close

close the currently open log; is automatically called, but users may want to call it themselves to force the respective file descriptor to be closed before the program is ended

returns: nothing

@StateChanging

b_flog_init [log file name] [header callback function] [log reduction lines]

flog/b_flog_init

Initialize this log writer. This function **must** be called before any others.

[log file name]: name of the log file to write to; special files such as /dev/stdout, /dev/stderr (default), /dev/tcp, /dev/udp are supported if your bash version supports them; the file doesn't need to exist, but directories above it must exist

[header callback function]: optional name of the function to be called whenever a new log entry is generated; the function must be defined as follows:

```
[header callback function] [severity]
[severity]: see [b_flog_log](#b_flog_log)
returns: the full header meant to be used for the current moment in time
         with the given severity (without knowing the message details)
         and sets a non-zero exit code on errors; errors may cause the
         message to be logged without header
```

[log reduction lines]: if set to a positive integer, reduce the log file approximately to that number of lines during logging (default: 3000) - see b_flog_setLogReductionLinesApprox for details; this option has no effect on non-file outputs (stdout, network output, ...)

returns: sets a non-zero exit code on errors and may exit the script

@StateChanging

@B_E

b_flog_log [message] [severity]

flog/b_flog_log

Log the given message with the given optional severity.

[message]: message to log

[severity]: users may pass arbitrary numbers or even Strings here, but it is recommended to stick to the priorities defined in $BLIB_FLOG_SEV (default: ${B_FLOG_SEV["info"]})

returns: sets a non-zero exit code on errors and may exit the script

@B_E

b_flogErrorHandler [error out] [print stderr] [print stack trace] [print log errors] [log stack trace] [severity]

flog/b_flogErrorHandler

An alternative to b_defaultErrorHandler which will log program errors using flog.

[error out]: see b_defaultErrorHandler (default: 0)

[print stderr]: see b_defaultErrorHandler (default: 1 / log only)

[print stack trace]: see b_defaultErrorHandler (default: 1 / log only)

[print log errors]: whether or not to print errors related to the logging itself to stderr (default: 0 / print)

[log stack trace]: whether or not to log the stack trace (default: 1 / do not log it)

[severity]: the severity to use for all errors, defaults to ${B_FLOG_SEV[“critical”]}

returns: see b_defaultErrorHandler

b_flog_getDeps

flog/b_flog_getDeps

Get the dependencies of this module.

returns: newline-separated list of dependencies of this module

b_flog_getDateFormat

flog/b_flog_getDateFormat

Get the date format used for the header by this log writer (see “man date” for explanations).

returns: see above

b_flog_setDateFormat [format string]

flog/b_flog_setDateFormat

Set the date format used for the header by this log writer (see “man date” for explanations).

returns: nothing

@StateChanging

b_flog_getLogReductionLinesLowerBound

flog/b_flog_getLogReductionLinesLowerBound

Get the number of lines that the log file will at least have after a log file reduction.

returns: see above

b_flog_getLogReductionLinesUpperBound

flog/b_flog_getLogReductionLinesUpperBound

Get the maximum number of lines that the log file will have before it is reduced.

returns: see above

b_flog_setLogReductionLinesLowerBound [bound]

flog/b_flog_setLogReductionLinesLowerBound

Set the number of lines that the log file will at least have after a log file reduction.

[bound]: number of lines to use for that bound

returns: nothing

@StateChanging

b_flog_setLogReductionLinesUpperBound

flog/b_flog_setLogReductionLinesUpperBound

Set the maximum number of lines that the log file will have before it is reduced.

[bound]: number of lines to use for that bound

returns: nothing

@StateChanging

b_flog_setLogReductionLinesApprox [line count]

flog/b_flog_setLogReductionLinesApprox

Set the number of average number of lines that the log file should have; counts <= 0 indicate no limit.

[line count]: reduce the log after reaching 1.2*[line count] lines to 0.8*[line count] lines

returns: nothing

@StateChanging

b_flog_getHeaderFunction

flog/b_flog_getHeaderFunction

Get the name of the header callback function that is used.

returns: see above

b_flog_setHeaderFunction [header function]

flog/b_flog_setHeaderFunction

Set the name of the header callback function to be used.

[header function]: name of the header function to use

returns: nothing

@StateChanging

Header Functions

b_flog_defaultHeader [severity]

flog/b_flog_defaultHeader

Default header callback function used with b_flog_init.

[severity]: the default header ignores the severity

returns: the default header meant to be used for the current moment in time

b_flog_headerDateSeverity [severity]

flog/b_flog_headerDateSeverity

An alternative to the default header callback function which appends the severity to the default header.

[severity]: see b_flog_init

returns: the default header with the severity appended

b_flog_headerDateScriptSeverity [severity]

flog/b_flog_headerDateScriptSeverity

An alternative to the default header callback function which appends the calling script and the severity to the default header.

[severity]: see b_flog_init

returns: the default header with the calling script and severity appended

fs

Collection of file and file system related functions.

0.3

Dependencies

wc
date
findmnt
mktemp
mount

Functions

b__fs__getDeps

fs/b_fs_getDeps

Get the dependencies of this module.

returns: newline-separated list of dependencies of this module

b__fs__isEmptyDir [dir]

fs/b_fs_isEmptyDir

Check whether the given directory is empty or non-existing. It is not checked whether the passed parameter is a file preventing a directory from being created.

[dir]: full path to the directory to check

returns: a zero exit code if the directory does not exist or is empty

b__fs__getLastModifiedInDays [file]

fs/b_fs_getLastModifiedInDays

Get the number of days since when a file was last modified.

[file]: Full path to the file to check.

returns: The time in days since the last modification. Sets a non-zero exit code on errors.

@B_E

b__fs__getLineCount [file]

fs/b_fs_getLineCount

Get the number of lines of the given file.

[file]: full path to a file

returns: the number of lines; a non-zero exit code is set on errors

@B_E

b__fs__waitForFile [file] [maximum time]

fs/b_fs_waitForFile

Sleep until the given file appears. The check interval is 1s.

[file]: full path to the file or directory to wait for

[maximum time]: maximum time in s to wait for the file to appear (default: forever)

returns: Sets a zero exit code if the file appeared and a non-zero exit code on a timeout.

b_fs_getMountpoints [device]

fs/b_fs_getMountpoints

Get all mountpoints for the given device.

[device]: Full path to the device (incl. /dev/) for which to obtain the mountpoints.

returns: A newline-separated list of mountpoints where the given device is mounted to. Sets a non-zero exit code if no such mountpoints were found.

b_fs_mountIfNecessary [device] [mount point]

fs/b_fs_mountIfNecessary

Mount the given device if it isn't already mounted.

[device]: Full path to the device (incl. /dev/) to mount.

[mount point]: Full path where to mount the device. If no mount point is specified, a /tmp/ mount point is chosen. Non-existing directories are created. Is ignored if another mount point already exists.

returns: The chosen mount point or a newline-separated list of existing mount points on success; sets a non-zero exit code on failure.

@B_E

b_fs_createLoopDeviceIfNecessary [file]

fs/b_fs_createLoopDeviceIfNecessary

Create a loop device for the given file if no old one exists. Usually requires root access rights.

[file]: File for which to create a loop device.

returns: Created loop device or previously used one (incl. /dev/). Sets a non-zero exit code, if no device could be created.

@B_E

b_fs_parseSize [string] [check flag]

fs/b_fs_parseSize

Parse human-readable file system sizes that include units.

[string]: A string denoting a file system size of the format [number][unit]. Unit may be one of KB 1000, K 1024, MB 1000*1000, M 1024*1024, GB 1000*1000*1000, G 1024*1024*1024, and so on for T, P. If no unit is provided, the number is assumed to denote bytes.

[check flag]: Check whether the result makes sense (default: 0/true). This check will make integer overflows less likely.

returns: The respective number of bytes meant with the given string. B_E is called on parsing errors.

@B_E

http

Collection of http related functions.

Copyright (C) 2018 David Hobach LGPLv3
0.2

Dependencies

curl

Global Variables

B_HTTP_CHECKURLS

http/B_HTTP_CHECKURLS

Each call to b_http_getOnlineStatus causes one of the URLs in this array to be visited.
It is recommended to pick a relatively large number of URLs with SSL support to remain relatively anonymous, if b_http_getOnlineStatus is called multiple times.
Defaults to the European/US Alexa Top 10 (which hopefully blends in to the masses).

Functions

b_http_getDeps

http/b_http_getDeps

Get the dependencies of this module.

returns: newline-separated list of dependencies of this module

b_http_rawUrlEncode [string]

http/b_http_rawUrlEncode

Encode the given string according to RFC 3986.

[string]: to encode

returns: Returns a string in which all non-alphanumeric characters except -_.~ have been replaced with a percent (%) sign followed by two hex digits. This is the encoding described in RFC 3986 for protecting literal characters from being interpreted as special URL delimiters, and for protecting URLs from being mangled by transmission media with character conversions (like some email systems). A non-zero exit code is set on errors.

@B_E

b_http_rawUrlDecode [string]

http/b_http_rawUrlDecode

Decode the given string encoded with b_str_rawUrlEncode or an equivalent function.

[string]: to decode

returns: The literal string with all hex characters replaced; a non-zero exit code is set on errors.

b_http_getOnlineStatus [timeout]

http/b_http_getOnlineStatus

Find out whether we are online or not by attempting an http connection.
One of B_HTTP_CHECKURLS is possibly visited during the process.

[timeout]: Timeout in seconds for hanging checks (default: 5).

returns: 0, if we're online, 1 if only DNS works, 2 if neither DNS nor http(s) worked, 3 if the check timed out; B_E will be called if the status cannot be determined.

@B_E

ini

Stateful ini reader for bash.

Currently only a single file per instance of this library/thread is kept in memory, but you can read multiple files one after another or in multiple threads.

Implementation Specifics:

- names/keys & values are case sensitive
- comment lines may start with ; or #
- whitespace lines are ignored
- duplicate names may result in undefined behaviour (usually the second will override the first)
- all characters following the = are considered part of the value (incl. whitespace); whitespace before and after the value may be trimmed by the getters though (check their description)
- values are not interpreted (e.g. quotes, escape characters, ...)
- whitespace around keys and around section qualifiers is ignored

0.5

Dependencies

no dependencies

Functions

b_ini_getDeps

ini/b_ini_getDeps

Get the dependencies of this module.

returns: newline-separated list of dependencies of this module

b_ini_read [ini file]

ini/b_ini_read

read the given ini file and keep it in thread-local memory so that subsequent calls to the b_ini_get functions will return the values from the ini file; subsequent calls to this function will update the internal state to represent the file last read in this thread

[ini file]: path to the ini file to read

returns: an error message on errors and sets a non-zero exit code on errors

@StateChanging

@B_E

b_ini_get [name] [section]

ini/b_ini_get

get the value for the ini entry with the given name as String in raw format

[name]: name/key of the ini entry to retrieve

[section]: section where to look for the entry with the given name (default: without section)

returns: value of the ini entry matching exactly the given section and name incl. any whitespace; a non-zero exit code is set if such an entry wasn't found or another error occurred

b_ini_getString [name] [section]

ini/b_ini_getString

get the value for the ini entry with the given name as String and remove all whitespace around the returned String

[name]: name/key of the ini entry to retrieve

[section]: section where to look for the entry with the given name (default: without section)

returns: value of the ini entry matching exactly the given section and name excl. any whitespace around; a non-zero exit code is set if such an entry wasn't found or another error occurred

b_ini_getInt [name] [section]

ini/b_ini_getInt

get the value for the ini entry with the given name as integer

[name]: see b_ini_get

[section]: see b_ini_get

returns: see b_ini_get; additionally it is checked whether the return value is an integer (if not, a non-zero exit code is set and the return value is undefined)

b_ini_getBool [name] [section]

ini/b_ini_getBool

get the value for the ini entry with the given name as boolean

[name]: see b_ini_get

[section]: see b_ini_get

returns: see b_ini_get; 0 is returned via echo for true, 1 for false; the exit code indicates a potential error during parsing or a missing entry (and *not* true/false)

multithreading/ipcm

An inter-process map implementation.

The map is available to all running processes during a single boot session.

Both reading and writing to the map can be done from any number of processes.

Overall Features	
# readers	multiple
# writers	multiple
read consistency	always
write consistency	always
blocking	on writes

0.2

Dependencies

no dependencies

Imports

multithreading/ipcv

multithreading/mtx

Functions

b_ipcm_getDeps

multithreading/ipcm/b_ipcm_getDeps

Get the dependencies of this module.

returns: newline-separated list of dependencies of this module

b_ipcm_setNamespace [namespace]

multithreading/ipcm/b_ipcm_setNamespace

Set the common process namespace to use. All processes inside the same namespace share a common state.
It is recommended to call this function a single time before any other functions of this module. Otherwise the default namespace, which may include unrelated processes, is used.

[namespace]: Name of the namespace to set.

returns: Errors out, if the name is unacceptable.

@StateChanging

@B_E

b_ipcm_getNamespace

multithreading/ipcm/b_ipcm_getNamespace

Retrieve the currently used namespace.

returns: The currently used namespace.

b_ipcm_change [key] [change function] [maximum time]

multithreading/ipcm/b_ipcm_change

Change the given key/value combination inside the map in a thread-safe way. May wait for changes done by other processes.

[key]: A global unique identifier for the given value.

[change function]: Name of the function to execute the change of the value. It will be called with the current key as first parameter and the current value as second. It is expected to print the new value to set for this key. A non-zero exit code will cause b_ipcm_change to abort the change.

[maximum time]: maximum time in ms to wait for other processes to complete their operation (default: -1 = indefinitely)

returns: A zero exit code and prints the new value, if the change succeeded. An exit code of B_RC+1 indicates that the change function returned a non-zero exit code. B_E is called otherwise.

@B_E

b_ipcm_get [key] [fallback]

multithreading/ipcm/b_ipcm_get

Retrieve the data found at the given key.

[key]: A global unique identifier for the data to retrieve.

[fallback]: Data to return if nothing was found for the given key (default: empty).

returns: Sets a zero exit code and returns the data found on success. If no data was found, the fallback data is returned and a zero exit code is set. B_E is called on errors.

b_ipcm_unsetNamespace [namespace] [maximum time]

multithreading/ipcm/b_ipcm_unsetNamespace

Unsets the given namespace and all keys stored within it.
It is recommended to call this function when all processes finished their work.

[namespace]: The namespace to unset (default: the current namespace).

[maximum time]: maximum time in ms to wait for other processes to complete their operation (default: 0 = indefinitely)

returns: Sets a zero exit code only upon successful removal. Otherwise B_E is triggered.

@B_E

multithreading/ipcv

Provides means for inter-process communication (ipc) via global bash variables (v).

This implementation uses shared memory, i.e. it should be reasonably fast.

Only a single process or thread is assumed to be writing (i.e. use b_ipcv_save) a variable at a time and multiple processes may read it (using e.g. b_ipcv_load). If you need to write a single variable from multiple processes, please consider using the multithreading/mtx module or similar locking means in combination with this module (or just multithreading/ipcm).

Overall Features	
# readers	multiple
# writers	single
read consistency	always
write consistency	only for one writer
blocking	never

0.3

Dependencies

findmnt
mkdir
mktemp
mv
rm

Functions

b_ipcv_getDeps

multithreading/ipcv/b_ipcv_getDeps

Get the dependencies of this module.

returns: newline-separated list of dependencies of this module

b_ipcv_save [namespace] [var name 1] .. [var name n]

multithreading/ipcv/b_ipcv_save

Save the current values of the given variables so that they are made available for other processes under the given namespace.

Please note that each variable is saved atomically, but individually. I.e. if you need multiple values to be updated at the same time, please use a single variable (e.g. a map).

[namespace]: Name for a common group under which the given variables should be saved. The combination of [namespace] and [variable name] must be a unique identifier across all processes running on the system.

[var name i]: The name of the global variable to make accessible for other processes. An arbitrary number of variable names can be specified.

returns: Nothing, but a non-zero exit code indicates failed variable save attempts. A failed save attempt also triggers B_E.

@B_E

b_ipcv_load [namespace] [var name 1] .. [var name n]

multithreading/ipcv/b_ipcv_load

Load the given variables from the given namespace into the current process context.

[namespace]: Name of the group under which the variable was saved with b_ipcv_save. Must exist.

[var name i]: Name of the variable to load. Multiple names can be specified.

returns: A non-zero exit code indicates the number of variables that could not be loaded unless some unexpected error occurred and B_E is triggered. Please note that a failed load attempt/unavailable variable does generally not trigger B_E.

@B_E

b_ipcv_loadNamespace [namespace] [check existence]

multithreading/ipcv/b_ipcv_loadNamespace

Load all variables that can be loaded for the given namespace into the current process context.

[namespace]: Name of the group for which to load all available variables.

[check existence]: Whether or not to make sure that the namespace to load exists (default: 0/check). Otherwise non-existing namespaces will not cause an error.

returns: Sets a zero exit code on success. Failing to load any single available variable will always trigger B_E.

@B_E

b_ipcv_unset [namespace] [var name 1] .. [var name n]

multithreading/ipcv/b_ipcv_unset

Unset/Remove the given variables from the global namespace.
Please note that the variables will remain set in your current process context, if they were set before. Use the standard bash `unset` for that.

[namespace]: Group where the given variables belong to.

[var name i]: Name of the variable to remove. Multiple may be specified.

returns: The number of variables which could not be unset. B_E is not triggered for these.

@B_E

b_ipcv_unsetNamespace [namespace]

multithreading/ipcv/b_ipcv_unsetNamespace

Remove the given global namespace and all variables it contains.
Please note that the variables will remain set in your current process context, if they were set before. Use the standard bash `unset` for that.

[namespace]: To remove.

returns: Sets a zero exit code only upon successful removal. Otherwise B_E is triggered.

@B_E

multithreading/mtx

Collection of mutex related functions.

Mutex: Only a single process may have it at any point in time.
Lock: A specific maximum number of processes may have it at any point in time.

0.3

Dependencies

mkdir
touch
sleep
rm
rmdir
cat
mktemp

Imports

proc

Functions

b_mtx_getDeps

multithreading/mtx/b_mtx_getDeps

Get the dependencies of this module.

returns: newline-separated list of dependencies of this module

b__mtx__setSleepTime [ms]

multithreading/mtx/b_mtx_setSleepTime

Sets the time to sleep for this module whenever active polling is done (default: 500).

[ms]: time in miliseconds between active polling requests for e.g. mutexes done by this module; must be an integer

returns: Nothing, always sets a zero exit code.

b__mtx__getSleepTime

multithreading/mtx/b_mtx_getSleepTime

Gets the time to sleep for this module whenever active polling is done.

returns: The currently set time to sleep in ms.

b__mtx__create [base dir]

multithreading/mtx/b_mtx_create

Allocate a new mutex without claiming it (use b_mtx_try for that).

[base dir]: Path to an *existing* directory where to store the mutex (default: not specified). By default this module will pick a temporary location. If you need a mutex that persists across reboots, please set a directory that persists across reboots here. The path should point to a local, non-network file system destination. The module must be able to create remove files or directories there at will.

returns: A string identifying the mutex (mutex ID). Sets a non-zero exit code on errors.

@B_E

b__mtx__release [mutex] [block ID]

multithreading/mtx/b_mtx_release

Release the given mutex so that it can be used by other block IDs/threads.

[mutex]: A mutex obtained via b_mtx_create.

[block ID]: The block ID for which to release the mutex (default: $$).

returns: Sets a non-zero exit code if the mutex could not be removed as another process is blocking it and a zero exit code on successful removal.

b_mtx_forceRelease [mutex]

multithreading/mtx/b_mtx_forceRelease

Release the given mutex so that it can be used by other blockIDs/threads. Warning: This function can remove mutexes from other threads and should generally *only* be used for the removal of mutexes which are known to be stale by the calling application.

[mutex]: A mutex obtained via b_mtx_create.

returns: Nothing and sets a zero exit code.

b_mtx_pass [mutex] [block ID]

multithreading/mtx/b_mtx_pass

Pass a blocked mutex to another block ID (i.e. change the block ID of the given mutex).
You should only do this if you currently own the mutex and the new process is ready to take over.

[mutex]: A mutex obtained via b_mtx_create.

[block ID]: The block ID to set for the given mutex.

returns: Sets a zero exit code on success and a non-zero exit code otherwise.

@B_E

b_mtx_try [mutex] [block ID] [claim stale] [claim own]

multithreading/mtx/b_mtx_try

Attempt to obtain the given mutex. Return immediately even if it cannot be obtained.

[mutex]: A mutex obtained via b_mtx_create. You may also use a static and otherwise unused directory path as mutex and share it across all relevant processes.

[block ID]: The ID to use by which to block (default: running (sub)shell process id $$). This should be the process ID of the process attempting to obtain the mutex or you should know what you're doing. If you're in a subshell that should deploy a mutex against other subshells, store their $BASHPID and call the function with that.

[claim stale]: If set to 0, claim the mutex even if it is still blocked by some other process, but that process isn't running anymore. If set to 1 (default), the function returns without obtaining the mutex. In general this should only be used in situations where a mutex has a high probability of being stale (e.g. application start).

[claim own]: If set to 0 (default), claim the mutex if it appears to be blocked by the provided block ID. If set to 1, consider it blocked even then.

returns: The function incl. parameters to execute to remove the mutex if it was obtained and an error message stating the reason otherwise. The provided function *should* be called as part of an exit trap of the calling script or via eval. Sets an exit code of 0, if the mutex was obtained. An exit code of 1 is set, if the mutex was blocked and another non-zero exit code if some other error occurred (the mutex might be blocked even then).

Example code:

```
local mutex=""
local mutexRet=""
mutex="$(b_mtx_create)" || { B_ERR="Failed to create a mutex." ; B_E }
mutexRet="$(b_mtx_try "$mutex")" \
|| { B_ERR="Failed to obtain the mutex $mutex. Reason: $mutexRet" ; B_E }
#assuming the mutex is only meant to be removed after full
#execution of the script:
trap "$mutexRet" EXIT
#direct removal:
#b_mtx_release "$mutex"
```

b_mtx_waitFor [mutex] [block ID] [claim stale] [maximum time]

multithreading/mtx/b_mtx_waitFor

Wait for the given mutex to become available. This will block script execution.

[mutex]: see b_mtx_try

[block ID]: see b_mtx_try

[claim stale]: see b_mtx_try

[maximum time]: maximum time in ms to wait for the mutex to become available (default: -1 = indefinitely)

returns: see b_mtx_try

multithreading/multiw

Allow multiple processes to write to a *virtual* file at the same time without causing write inconsitencies (written data from each process mangled with each other).

This is achieved by keeping one file per process and relies on the assumption that both replacing and reading a symlink on your Linux distribution is atomic.

In order for this to work, all write operations must go through this module.

Currently reading only returns the data written by the process which wrote last. If you need some sort of appending, it makes more sense to deploy a mutex using e.g. the multithreading/mtx module.

Overall Features	
# readers	multiple
# writers	multiple
read consistency	partial, last writer wins
write consistency	always
blocking	never

0.2

Dependencies

shuf
mv
ln
stat
rm

Functions

b_multiw_getDeps

multithreading/multiw/b_multiw_getDeps

Get the dependencies of this module.

returns: newline-separated list of dependencies of this module

b_multiw_setMaxHangTime [seconds]

multithreading/multiw/b_multiw_setMaxHangTime

Set the maximum time that a process is expected to hang between two instructions. This is relevant for various internal guarantees.

[seconds]: Time in seconds that a process hangs at most.

returns: Nothing.

@StateChanging

b_multiw_getMaxHangTime

multithreading/multiw/b_multiw_getMaxHangTime

Get the maximum time that a process is expected to hang between two instructions. This is relevant for various internal guarantees.

returns: Time in seconds.

b_multiw_write [file path]

multithreading/multiw/b_multiw_write

Write all data lying in stdin to the given *virtual* file in a thread-safe way.

[file path]: Full path to the virtual file to write to. Must not be a regular file (but may not exist).

returns: A zero exit code, if the write operation was successful and a non-zero exit code otherwise.

@B_E

b_multiw_remove [file path]

multithreading/multiw/b_multiw_remove

Remove the given virtual file *and all of its revisions.*
This function should only be called when all processes finished reading and writing. It is recommended to use it over the standard Linux `rm` as the latter will leave remnants behind.

returns: A zero exit code on success.

@B_E

os/osid

Functions for operating system identification.

0.2

Dependencies

no dependencies

Functions

b_osid_getDeps

os/osid/b_osid_getDeps

Get the dependencies of this module.

returns: newline-separated list of dependencies of this module

b_osid_init [force]

os/osid/b_osid_init

[force]: if set to 0, force an init even if it would otherwise not be necessary (default: 1 - only initialize if it didn't happen before)
Initialize the osid module. It should normally *not* be necessary to call this function directly, but it will be called by the osid module internally as needed.

returns: May error out and set a non-zero exit code on failures.

b_osid_isDebian

os/osid/b_osid_isDebian

Check whether the OS running this function is a Debian Linux.

returns: Sets a zero exit code if the check returns true. Does not print any output.

b_osid_isDebianLike

os/osid/b_osid_isDebianLike

Check whether the OS running this function is a Debian Linux or one of its derivatives (e.g. ubuntu).

returns: Sets a zero exit code if the check returns true. Does not print any output.

b_osid_isOpenSuse

os/osid/b_osid_isOpenSuse

Check whether the OS running this function is a OpenSUSE.

returns: Sets a zero exit code if the check returns true. Does not print any output.

b__osid__isFedora

os/osid/b_osid_isFedora

Check whether the OS running this function is a Fedora Linux.

returns: Sets a zero exit code if the check returns true. Does not print any output.

b__osid__isCentOS

os/osid/b_osid_isCentOS

Check whether the OS running this function is a CentOS.

returns: Sets a zero exit code if the check returns true. Does not print any output.

b__osid__isRedHat

os/osid/b_osid_isRedHat

Check whether the OS running this function is a RedHat Linux.

returns: Sets a zero exit code if the check returns true. Does not print any output.

b__osid__isUbuntu

os/osid/b_osid_isUbuntu

Check whether the OS running this function is an Ubuntu Linux.

returns: Sets a zero exit code if the check returns true. Does not print any output.

b__osid__isFedoraLike

os/osid/b_osid_isFedoraLike

Check whether the OS running this function is a Fedora Linux or one of its derivatives (e.g. CentOS, Red Hat, Qubes OS).

returns: Sets a zero exit code if the check returns true. Does not print any output.

b_osid_isQubesDom0

os/osid/b_osid_isQubesDom0

Check whether the OS running this function is a Qubes OS in dom0.

returns: Sets a zero exit code if the check returns true. Does not print any output.

b_osid_isQubesVM

os/osid/b_osid_isQubesVM

Check whether the OS running this function is a Qubes OS in a VM.

returns: Sets a zero exit code if the check returns true. Does not print any output.

os/qubes4/dom0

Collection of functions supporting scripting in Qubes OS 4.x dom0.

Important: Whenever you parse output from VMs to dom0, you **must** be extra careful and assume it totally untrusted as parsing bugs are a plausible attack vector for compromised VMs. Passing data to potentially compromised VMs of course also exposes that data's confidentiality.

0.4

Dependencies

qvm-run
qvm-block
qvm-ls
qvm-copy-to-vm
qvm-prefs
qvm-shutdown
qvm-check
qubes-prefs
dirname
basename
mktemp
ps
sleep
cat
timeout

dd
tee
xxd
python3
kill

Imports

fs

proc

types

Functions

b_dom0_getDeps

os/qubes4/dom0/b_dom0_getDeps

Get the dependencies of this module.

returns: newline-separated list of dependencies of this module

b_dom0_qvmRun [parameter 1] ... [parameter n]

os/qubes4/dom0/b_dom0_qvmRun

A wrapper for qvm-run which sets reasonable defaults for shell scripting and applies various fixes.

Most calls to qvm-run should be made via this function rather than interacting with qvm-run directly as the Qubes OS qvm-run was designed with interactive shell usage in mind whereas this wrapper is intended for bash developers.

Particular features:

- a certain set of reasonable default parameters is used: -p -q -n -u root
- -n was set as auto-starting VMs during Bash scripting can heavily influence the user experience (imagine the VM being shut down manually by the user whilst a bash script is running –> constant restarts)
- stdin is redirected to /dev/null by default to avoid potential security implications (accidental reads from dom0 stdin passed to a VM); this can be overriden using –stdin
- stdout has the VM output and the exit code is the one of the VM
- distinguished exit conditions (executed command failed vs. qvm-run failed)
- workarounds for known Qubes bugs wrt qvm-run may be implemented here (e.g. qubes issues #3083, #4476, #4633 in the past)

Please note that calling this function will make your script wait for the execution of the commands in the client VM.

Wherever possible, this function should be combined with b_silence as the VM output shouldn't be trusted. Otherwise please keep in mind that **both** stdout *and* stderr may have untrusted output which may even contain binary data. In order to validate against binary data you can e.g. use b_types_parseString.

[parameters]: Any parameters supported by qvm-run. If you pass -a, the default -n will be overriden. If you pass -u, the default root user is overriden. If you pass -v, -q will be overriden. If you pass –stdin, even stdin is passed to qvm-run. -p can be overriden by using /dev/null redirection. Only the short parameter versions are supported.

returns: Sets the exit code of qvm-run and prints its output. May error out using B_E if qvm-run itself fails.

@B_E

b_dom0_getDispVMs

os/qubes4/dom0/b_dom0_getDispVMs

Get a list of all currently existing disposable VMs.

[returns]: The currently existing disposable VMs as newline-separated list.

@B_E

b_dom0_startDispVM [template]

os/qubes4/dom0/b_dom0_startDispVM

Start a dispVM from the given template in the background and return its name. The disposable VM will remain started until it is shut down. If you only wish to execute a single command, please use b_dom0_qvmRun with the –dispVM parameter.
It may take a while for this function to obtain the name of the dispVM.

[template]: The template to use for the dispVM. If no template is specified, use the default Qubes template.

returns: Name of the dispVM that was started and sets a zero exit code on success. This function may error out.

@B_E

b_dom0_execIn [vm] [file] [user]

os/qubes4/dom0/b_dom0_execIn

Execute the file as bash code in the given VM and wait for it to finish.

See b_dom0_qvmRun for various notes and words of caution.

[vm]: Name of the VM where to execute the given string. The VM is assumed to be started.

[file]: Bash file to execute in the given VM.

[user]: user as which to execute the bash file (default: root)

returns: Whatever the executed Bash code prints in the VM to stderr or stdout; the status code is set to the one of the executed Bash code on success (0). Non-zero exit codes and error messages may come from both this function as well as the code executed in the given VM.

@B_E

b_dom0_execStrIn [vm] [string] [user]

os/qubes4/dom0/b_dom0_execStrIn

Execute the String as bash code in the given VM and wait for it to finish.

Convenience wrapper for b_dom0_execIn.

See b_dom0_qvmRun for various notes and words of caution.

[vm]: see b_dom0_execIn

[string]: Bash String to execute in the given VM.

[user]: see b_dom0_execIn

returns: see b_dom0_execIn; B_E is *not* called if the executed command returns an error

@B_E

b_dom0_execFuncIn [vm] [user] [function] [param 1] .. [param p] - [module dep 1] .. [module dep n] - [function dep 1] .. [function dep d]

os/qubes4/dom0/b_dom0_execFuncIn

Execute the Bash function in the given VM and wait for it to finish.

Convenience wrapper for b_dom0_execIn.

See b_dom0_qvmRun for various notes and words of caution.

[vm]: see b_dom0_execIn

[user]: see b_dom0_execIn

[function]: Name of the function as it is declared in the current scope.

[param p]: An arbitrary number of function parameters.

[-]: A dash as separator character between the various parameters.

[module dep i]: Names of the modules required by the function. They do not need to be imported by the function itself.

[function dep j]: An arbitrary number of functions that need to be added in order to satisfy the dependencies of the function to call (e.g. if function A is meant to be called, but uses function B internally, you'll have to pass B as one of its dependencies). Dependencies that can be found in added modules must *not* be added.

returns: see b_dom0_execIn; B_E is *not* called if the executed command returns an error

@B_E

b_dom0_waitForFileIn [vm] [file] [maximum time]

os/qubes4/dom0/b_dom0_waitForFileIn

Convenience wrapper for b_fs_waitForFile.

[vm]: VM where to execute.

@B_E

b_dom0_isMountedIn [vm] [device]

os/qubes4/dom0/b_dom0_isMountedIn

Check whether the device is mounted in the given VM.

[vm]: VM where to execute.

[device]: Full path to the device (incl. /dev/) to check.

returns: Sets a zero exit code if the device is mounted in the VM; a non-zero exit code means that it's either not mounted or some other error occurred.

@B_E

b_dom0_mountIfNecessary [vm] [device] [mount point]

os/qubes4/dom0/b_dom0_mountIfNecessary

Mount the given device in the target VM if it isn't already mounted there. Actually a wrapper for b_fs_mountIfNecessary.

[vm]: VM where to execute.

[device]: Full path to the device (incl. /dev/) to mount.

[mount point]: Full path where to mount the device. If no mount point is specified, a /tmp/ mount point is chosen. Non-existing directories are created. Is ignored if another mount point already exists.

returns: The chosen mount point or a newline-separated list of existing mount points on success; sets a non-zero exit code on failure. As these strings are returned from the VM, extra care must be taken when parsing them.

@B_E

b_dom0_createLoopDeviceIfNecessary [vm] [file]

os/qubes4/dom0/b_dom0_createLoopDeviceIfNecessary

Create a loop device for the file in the given VM if no old one exists. Actually a wrapper for b_fs_createLoopDeviceIfNecessary.

This usually requires root privileges.

[vm]: VM where to execute.

[file]: File for which to create a loop device.

returns: Created loop device or previously used one (incl. /dev/). Sets a non-zero exit code, if no device could be created.

@B_E

b_dom0_copy [dom0 file] [target VM] [target VM dir] [overwrite] [parent dir]

os/qubes4/dom0/b_dom0_copy

Grab a file or directory in dom0 and push it to the given file path in the target VM.

[dom0 file]: location of the dom0 file or directory to read from, assumed to exist

[target VM]: VM to write to, assumed to exist. Must be started.

[target VM dir]: full path to the parent directory in the target VM to copy the file or directory to; non-existing parent directories are created; the name is taken from the name of the file/directory in dom0

[overwrite]: Whether or not to overwrite an existing [destination file] (default: 0 = overwrite).

[parent dir]: Set this to 0, if the [target VM dir] is the target *parent* directory (default) and to 1 if it includes the target file or folder *name* as last element.

returns: Sets an exit code of 0, if everything went fine, and a non-zero exit code otherwise.

@B_E

b_dom0_crossCopy [source VM] [source file] [target VM] [target VM dir] [overwrite] [parent dir]

os/qubes4/dom0/b_dom0_crossCopy

Cross copy a file or directory from one VM to another, initiated by dom0. No user prompt is displayed.

[source VM]: Where to copy the source file from. Must be started.

[source file]: The file or directory to copy.

[target VM]: Where to copy the file to. Must be started.

[target VM dir]: full path to the parent directory in the target VM to copy the file or directory to; non-existing parent directories are created; the name is taken from the name of the file/directory in dom0

[overwrite]: Whether or not to overwrite an existing [destination file] (default: 0 = overwrite).

[parent dir]: Set this to 0, if the [target VM dir] is the target *parent* directory (default) and to 1 if it includes the target file or folder *name* as last element.

returns: Sets an exit code of 0, if everything went fine, and a non-zero exit code otherwise.

@B_E

b_dom0_ensureRunning [vm]

os/qubes4/dom0/b_dom0_ensureRunning

Start the given VM if needed.

[vm]: The VM to start.

returns: Sets a zero exit code, if the VM was successfully started or was running and a non-zero exit code otherwise. B_E will only be called for internal errors.

@B_E

b_dom0_isRunning [vm]

os/qubes4/dom0/b_dom0_isRunning

Check whether the given VM is running *and* fully operational / not hanging / not booting.

In contrast e.g. qvm-check –running [vm] appears to return true for VMs which are currently booting; qvm-ls doesn't check whether the OS of a VM is hanging. This should *not* be checked too often as it may be expensive.

[vm]: The VM to check.

returns: Sets a zero exit code, if the VM is running and a non-zero exit code otherwise. A non-zero exit code may e.g. also indicate that the VM doesn't exist. B_E will only be called for internal errors.

@B_E

b_dom0_exists [vm]

os/qubes4/dom0/b_dom0_exists

Check whether the given VM exists.

[vm]: The VM to check.

returns: Sets a zero exit code, if the VM exists a non-zero exit code otherwise.

b_dom0_openCrypt [vm] [device] [mapper name] [rw flag] [mount point] [key file] [type]

os/qubes4/dom0/b_dom0_openCrypt

In the given VM, open the given crypto device with dm-crypt and mount it to the mount point.

[vm]: The VM where to open the crypto device.

[device]: Full path to the device (incl. /dev/) to open.

[mapper name]: The name to assign to the decrypted version of the crypto block device. The created decrypted device will be found at */dev/mapper/[mapper name]*.

[rw flag]: 0=open read-write, 1=open read-only (default: 0)

[mount point]: Where to mount the decrypted data to. Non-existing directories will be created. If no mount point is specified, it will not be mounted (default).

[key file]: Full vm path to the key to use for decryption. If none is specified, password-based decryption is assumed and stdin will be read to obtain the password.

[type]: Type of encryption container to open: plain|luks|loopaes|tcrypt (default: luks)

returns: nothing (except for user interaction prompts if no key file is provided), but sets a non-zero exit code on errors

@B_E

b_dom0_closeCrypt [vm] [mapper name] [mount point]

os/qubes4/dom0/b_dom0_closeCrypt

Close a crypto device opened with b_dom0_openCrypt.

[vm]: The VM where to close the crypto device.

[mapper name]: The name used when it was opened.

[mount point]: If the decrypted data is mounted inside the [vm], please specify the mount point here so that it can be unmounted before closing the device. Otherwise the function will attempt to close the device without unmounting (likely to fail).

returns: nothing, but sets a non-zero exit code on errors

@B_E

b_dom0_parseQvmBlock [variable name] [input]

os/qubes4/dom0/b_dom0_parseQvmBlock

Parse data from `qvm-block ls` to an associative array.
The indices of the associative array will be of the format *[counter]_[field]* (counters run from 0 (inclusive) to max (exclusive)).
The special index "max" is equal to the number of lines. It can be used for iterations over the map.
Currently supported [field] values: backend|device id|id|description|used by|read-only|frontend-dev

[variable name]: Name of the associative array to use as output.

[input]: Optional output from a previous `qvm-block ls` call; if none is specified, this function will execute the call and use its output.

returns: A string specifying an associative array in bash syntax. You can eval that string to obtain all relevant data or use the more convenient b_dom0_getQvmBlockInfo. On errors B_E is called.

@B_E

b_dom0_getQvmBlockInfo [map] [retrieve field] [filter field 1] [filter value 1] .. [filter field n] .. [filter value n]

os/qubes4/dom0/b_dom0_getQvmBlockInfo

Convenience function to retrieve information from the output of b_dom0_parseQvmBlock. Searches for the given filter values in the given fields and retrieves the first field

value matching all filters.
More simple, but less flexible than b_dom0_parseQvmBlock.

[map]: Optional output from a previous call to b_dom0_parseQvmBlock. If none is specified, this function will internally call b_dom0_parseQvmBlock.

[retrieve field]: Name of the field to retrieve.

[filter field i]: Name of any field supported by b_dom0_parseQvmBlock.

[filter value i]: The value to search for in [filter field i] (equality check).

returns: The value of the requested field and sets a zero exit code on success. Sets a non-zero exit code if no matching value could be found. On errors B_E is called.

@B_E

b_dom0_attachFile [dom0 file] [target VM] [rw flag]

os/qubes4/dom0/b_dom0_attachFile

Attach the given file from dom0 (!) as block device to the target VM.
The function may attempt to acquire root privileges (and thus display a password prompt).

[dom0 file]: Full path to the file *in dom0* to attach.

[target VM]: VM to attach the file to. Must be started.

[rw flag]: If set to 0, attaches the dom0 file in r/w (read-write) mode. If set to 1 (default), attaches the file in r/o (read only) mode.

returns: The full path to the device created in the target VM and sets a zero exit code on success. Otherwise a non-zero exit code is set.

@B_E

b_dom0_attachVMDisk [source VM] [target VM] [dom0 working folder] [rw flag]

os/qubes4/dom0/b_dom0_attachVMDisk

Attach the entire private disk image (private.img) of the source VM to the target VM.
Warning: This is contradictory to all Qubes principles and should only be done if you know exactly what you're doing. Qubes OS even has some countermeasures to prevent accidental use of this feature which are bypassed here.

[source VM]: Name of the VM whose private disk to attach to the target VM. *All data* of that VM will be shared with the target VM. Will be shut down as part of this function and must remain shut down as long as the disk is attached.

[target VM]: VM where to attach the disk as block device to. Must be started.

[dom0 working folder]: Path to a folder that this function may use at will to create or delete temporary data. Must be on the same drive as /var/lib/qubes (for example /tmp/ does *not* work) and should **exclusively** be used for calls to this function. Can safely be removed once your program finishes and the target VM is shut down.

[rw flag]: If set to 0, attaches the disk file in r/w (read-write) mode. If set to 1 (default), attaches the file in r/o (read only) mode.

returns: The full path to the device created in the target VM and sets a zero exit code on success. Otherwise a non-zero exit code is set.

@B_E

b_dom0_crossAttachDevice [source VM] [source device] [target VM] [rw flag]

os/qubes4/dom0/b_dom0_crossAttachDevice

Attach the given block device from the source VM to the target VM.
This is merely a convenience wrapper for `qvm-block attach`.

[source VM]: VM where the source file can be found.

[source device]: Device to attach to the [target VM].

[target VM]: VM to attach the device to. Must be started.

[rw flag]: If set to 0, attaches the [source deivce] in r/w (read-write) mode. If set to 1 (default), attaches the file in r/o (read only) mode.

returns: The full path to the device created in the target VM and sets a zero exit code on success. Otherwise a non-zero exit code is set.

@B_E

b_dom0_crossAttachFile [source VM] [source file] [target VM] [rw flag]

os/qubes4/dom0/b_dom0_crossAttachFile

Attach the given file from the source VM as block device to the target VM.

[source VM]: VM where the source file can be found.

[source file]: File to attach as block device.

[target VM]: VM to attach the file to. Must be started.

[rw flag]: If set to 0, attaches the [source file] in r/w (read-write) mode. If set to 1 (default), attaches the file in r/o (read only) mode.

returns: The full path to the device created in the target VM and sets a zero exit code on success. Otherwise a non-zero exit code is set.

@B_E

b_dom0_detachDevice [vm] [device]

os/qubes4/dom0/b_dom0_detachDevice

Attempts to detach the given device from the VM. This may fail if the VM is using the device and thus it is usually a better idea to just shut the VM down.

[vm]: VM from which to detach the device.

[device]: Full path to the device in the VM. E.g. the return values of b_dom0_crossAttachFile, b_dom0_attachFile or b_dom0_attachVMDisk.

returns: nothing, but sets a zero exit code on success

@B_E

b_dom0_enterEventLoop [callback function]

os/qubes4/dom0/b_dom0_enterEventLoop

Enter a blocking loop to react to Qubes OS events.

[callback function]: Name of the function to call for *every* Qubes OS event. Since the number of events may be high, the function should do appropriate filtering at high performance.
The callback function should be declared as follows:

```
callback_function_name [subject] [event name] [event info]
[subject]: The subject name Qubes OS provides. Usually the VM for which the
    event was reported. 'None' appears to mean 'dom0'.
[event name]: Name of the event for which the callback function was called.
[event info]: May contain additional information about the event (e.g.
              arguments).
returns:  Nothing. A non-zero exit code will abort further processing.
```

If you want to obtain an overview of Qubes OS events, please use the `qwatch` utility manually.

returns: Nothing, but uses the exit code of the last callback function execution as its own. B_E is only called on exceptional errors.

@B_E

proc

Collection of process and thread related functions.

0.1

Dependencies

tail
timeout

Functions

b_proc_getDeps

proc/b_proc_getDeps

Get the dependencies of this module.

returns: newline-separated list of dependencies of this module

b_proc_pidExists [pid]

proc/b_proc_pidExists

Check whether the given process ID exists on the system.

[pid]: process ID to check for existence (process exists)

returns: A zero exit code, if it exists and a non-zero exit code if it doesn't; this function attempts to check the existence of the given process across *all* users, but it cannot guarantee correctness if the user running this script has very low privileges.

b_proc_waitForPid [pid] [maximum time]

proc/b_proc_waitForPid

Wait for the given process to exit. If it doesn't exist, exit immediately.

[pid]: process ID of the process to wait for

[maximum time]: maximum time in seconds to wait for the process to exit (default: 0 = indefinitely)

returns: Nothing, always sets a zero exit code. Use b_proc_pidExists if you need to know whether the process finished.

str

Collection of string related functions.

0.1

Dependencies

no dependencies

Functions

b__str__getDeps

str/b_str_getDeps

Get the dependencies of this module.

returns: newline-separated list of dependencies of this module

b__str__stripQuotes [string]

str/b_str_stripQuotes

Remove any single or double quotes around the given string.

[string]: string which might be enclosed in single or double quotes (' or ")

returns: [string] without the enclosed single or double quotes, if there were any; if none were found the original string is returned; the exit code is always zero

b__str__trim [string]

str/b_str_trim

remove any whitespace from around a string

[string]: string to trim

returns: [string] beginning and ending without whitespace; the exit code is always zero

tcolors

Defines some tput related constants. In order to change terminal colors you can then use something such as

```
echo "$(tput setaf ${B_TCOLORS[red]})This is red, \
$(tput setaf ${B_TCOLORS[blue]})this blue, $(tput sgr0)this normal."
```

tput can do a lot more than colors, see: man tput & man terminfo.

0.1

Dependencies

tput

Global Variables

B_TCOLORS

tcolors/B_TCOLORS

Global map for human readable colors to tput style color identifiers.
Currently supported values: black|red|green|yellow|blue|magenta|cyan|white

Functions

b_tcolors_getDeps

tcolors/b_tcolors_getDeps

Get the dependencies of this module.

returns: newline-separated list of dependencies of this module

traps

Collection of trap related functions.

0.2

Dependencies

no dependencies

Functions

b_traps_getDeps

traps/b_traps_getDeps

Get the dependencies of this module.

returns: newline-separated list of dependencies of this module

b_traps_getCodeFor [signal]

traps/b_traps_getCodeFor

Retrieve the current trap code / commands for the given signal.

returns: The current code and sets a zero exit code on success.

@B_E

b_traps_add [code] [signal] [tag] [append flag]

traps/b_traps_add

Add the given commands to the given trap signal.

[code]: Whatever should be added to the trap.

[signal]: Name of the signal to add the commands to.

[tag]: An optional *unique* marker for these commands so that they can be removed with b_traps_remove later on.

[append flag]: Whether to append the new commands to the end (0: default) or insert them in the beginning (1).

returns: Whatever the internal call to *trap* to set the new trap returns.

@B_E

b_traps_prepend [code] [signal] [tag]

traps/b_traps_prepend

Prepend the given commands to the ones currently existing for the given trap signal.

Convenience wrapper to b_traps_add with [append flag] set to 1.

[code]: see b_traps_add

[signal]: see b_traps_add

[tag]: see b_traps_add

returns: see b_traps_add

@B_E

b_traps_remove [signal] [tag]

traps/b_traps_remove

Remove the commands tagged with the given tag from the signal trap.

[signal]: Name of the signal to remove the commands from.

[tag]: The *unique* marker to identify the commands to be removed.

returns: Nothing, but sets a zero exit code on success. May error out if the tag isn't found or the internal trap call failed.

@B_E

types

Functions for data type checks and conversions.

Copyright (C) 2018 David Hobach LGPLv3
0.2

Dependencies

mktemp
mkfifo
rm
wc
strings
tee
head

Global Variables

B_TYPES_ENCODINGS

types/B_TYPES_ENCODINGS

Global map for human readable string encodings which can be used for b_types_parseString.
Currently supported values: 7-bit|8-bit|16-bit-bigendian|16-bit-littleendian|32-bit-bigendian|32-bit-littleendian
See the strings manpage for further explanations.

Functions

b__types__getDeps

types/b_types_getDeps

Get the dependencies of this module.

returns: newline-separated list of dependencies of this module

b__types__parseString [encoding]

types/b_types_parseString

Checks whether whatever is lying in stdin is a string (and not binary) and if so, prints it to stdout.

Important:

- bash has major issues whenever binary data is involved. For example equality checks may return undefined results. So whenever you are unsure as to whether a variable is a string or not, better pass it thorugh this function.
- The input is taken from *stdin* rather than as parameter as binary parameters may also cause issues (special bytes etc.).
- Even builtins such as `echo` do not necessarily play well with binary data. So it is recommended to pipe binary data through this function before further processing.

Examples:

```
#check a file
b_types_parseString < "/path/to/potential/binary" > /dev/null && echo "It is a string file

#read parts of a file as string
str="$(dd if="/path/to/another/file" bs=1 skip=8 | b_types_parseString)"
[ $? -eq 0 ] && echo "Found the following string: $str"
```

[encoding]: The encoding of the string lying in stdin. Use B__TYPES__ENCODINGS for this parameter. Defaults to ${B__TYPES__ENCODINGS[“7-bit”]}, which makes sense in 99% of all cases as scripts should use ASCII only anyway (when no user-interaction is involved) in order to remain portable. Keep in mind that bash also needs to support the target encoding in order to support further processing.

returns: The data as String, if the input data was found to be a String. If no String was found to be lying in stdin, the output is an undefined string and a non-zero exit code is set. B__E is only called on exceptional errors.

@B__E

b_types_isInteger [string]

types/b_types_isInteger

Check whether the given String is an integer (positive or negative) or not.

[string]: The string to check. If it may be binary data, please make sure to pass it through b_types_parseString first.

returns: Nothing, but sets a zero exit code if and only if the given string represents an integer.

tests/00_first.bats

This is not a real test, but can be used to execute some code *before* any bats tests are run.

Copyright (C) 2018 David Hobach LGPLv3
0.1

tests/arr.bats

Bats tests for the arr module.

Copyright (C) 2018 David Hobach LGPLv3
0.3

tests/blib.bats

Bats tests for blib main.

Copyright (C) 2018 David Hobach LGPLv3
0.5

tests/cdoc.bats

Bats tests for the cdoc module.

Copyright (C) 2018 David Hobach LGPLv3
0.3

tests/daemon.bats

Bats tests for the daemon module.

Copyright (C) 2019 David Hobach LGPLv3
0.3

tests/date.bats

Bats tests for the date module.

Copyright (C) 2018 David Hobach LGPLv3
0.3

tests/flog.bats

Bats tests for the flog library.

Copyright (C) 2018 David Hobach LGPLv3
0.3

tests/fs.bats

Bats tests for the arr module.

Copyright (C) 2018 David Hobach LGPLv3
0.3

tests/http.bats

Bats tests for the http module.

Copyright (C) 2018 David Hobach LGPLv3
0.3

tests/ini.bats

Bats tests for the ini module.

Copyright (C) 2018 David Hobach LGPLv3
0.3

tests/multithreading/ipcm.bats

Bats tests for the ipcm module.

Copyright (C) 2019 David Hobach LGPLv3
0.3

tests/multithreading/ipcv.bats

Bats tests for the ipcv module.

Copyright (C) 2018 David Hobach LGPLv3
0.3

tests/multithreading/mtx.bats

Bats tests for the arr module.

Copyright (C) 2018 David Hobach LGPLv3
0.3

tests/multithreading/multiw.bats

Bats tests for the multiw module.

Copyright (C) 2019 David Hobach LGPLv3
0.3

tests/os/osid.bats

Bats tests for the os/osid module.

Copyright (C) 2018 David Hobach LGPLv3
0.5

tests/os/qubes4/dom0.bats

Bats tests for the os/qubes4/dom0 module.

Important: This is *test* code and should not be used in production environments as quite often it is lacking checks wrt untrusted VM output from e.g. b_dom0_qvmRun. Developers should follow the standards outlined there for their projects.

Copyright (C) 2018 David Hobach LGPLv3
0.4

tests/proc.bats

Bats tests for the arr module.

Copyright (C) 2018 David Hobach LGPLv3
0.1

tests/str.bats

Bats tests for the str module.

Copyright (C) 2018 David Hobach LGPLv3
0.3

tests/tcolors.bats

Bats tests for the tcolors module.

Copyright (C) 2018 David Hobach LGPLv3
0.3

tests/test_common.bash

Some code and vars meant to be shared across bats tests.

Copyright (C) 2018 David Hobach LGPLv3
0.5

Global Variables

TEST_STATE

tests/test_common.bash/TEST_STATE

A map which can be used to create a persistent state across multiple tests.
By default, bats creates a new shell environment for each test it runs, resetting all changes to global variables.
The state can be managed with the load/save/clearBlibTestState functions below.

Functions

loadBlib

tests/test_common.bash/loadBlib

Loads blib for testing.

skipIfNoUserData

tests/test_common.bash/skipIfNoUserData

Skip the test if no user data was found.

skipIfNoUserData

tests/test_common.bash/skipIfNoUserData

Skip the test if pandoc is not installed.

skipIfNotQubesDom0

tests/test_common.bash/skipIfNotQubesDom0

Skip the test if we're not running inside Qubes OS dom0.

loadBlibTestState

tests/test_common.bash/loadBlibTestState

Load the TEST_STATE with the data that was saved last via saveBlibState. If you want to use TEST_STATE, call this function during test setup.

saveBlibState

tests/test_common.bash/saveBlibState

Save the current TEST_STATE to make it available for further tests.

clearBlibTestState

tests/test_common.bash/clearBlibTestState

Clears the current test state and removes its persistent files.

testGetterSetter [setter function] [value to set] [reset]

tests/test_common.bash/testGetterSetter

Executes the given setter function *in the current environment* and makes sure the respective getter function (assumed to have the same name with just a *get* instead of *set*) returns that value.

[setter function]: name of the setter function to call

[value to set]: value to set with the setter function

[reset]: if set to 0 (default), reset the value back to its original value after testing the setter function

returns: nothing, but errors out on test failures

startTimer

tests/test_common.bash/startTimer

Start a new time measurement window.

returns: nothing

endTimer

tests/test_common.bash/endTimer

Get the differencee in time in seconds since the last time startTimer was called.

returns: time difference in seconds

funcTimeout [timeout] [function] [args]

tests/test_common.bash/funcTimeout

Run the given function with a timeout inside a subshell.

[timeout]: Timeout in seconds after which to terminate the function.

[function]: The function to execute.

[args]: Function arguments.

returns: An exit code of 124, if the function timed out. Otherwise returns whatever the function returned.

runSL [commands]

tests/test_common.bash/runSL

A version of the bats `run` command which makes sure that the bash runtime state does not change after running the given commands (SL = stateless).
This *should* be the default method of executing tests for blib.
If the given commands are expected to change the state, use runSC instead. In particular the default bats `run` should almost never be used.
Also prints an identifier for easier debugging. The identifier starts at 1 on the first run call per test and increases with each further run call.
WARNING: As the bats `run` it runs inside a subshell. So don't expect changes to persist.

[commands]: The commands to run.

returns: whatever bats run returns

runSC [commands]

tests/test_common.bash/runSC

A version of the bats `run` command which ignore changes to the bash runtime state (SC = state changing).
If the given commands are expected to be stateless, use runSL instead. In particular the default bats `run` should almost never be used.
Also prints an identifier for easier debugging. The identifier starts at 1 on the first run call per test and increases with each further run call.
WARNING: As the bats `run` it runs inside a subshell. So don't expect changes to persist.

[commands]: The commands to run.

returns: whatever bats run returns

tests/traps.bats

Bats tests for the cdoc module.

Copyright (C) 2018 David Hobach LGPLv3
0.3

tests/types.bats

Bats tests for the types module.

Copyright (C) 2018 David Hobach LGPLv3
0.3

tests/user_test_data.bash.example

Example of a user_test_data file.
That file is meant for static bash varibles that must be set *by the user* (as they may e.g. be OS specific) in order for some tests to work.
If that file doesn't exist or could not be loaded, these tests may be skipped by blib.

Copyright (C) 2018 David Hobach LGPLv3
0.5

Global Variables

UTD_ONLINE

tests/user_test_data.bash.example/UTD_ONLINE

Whether or not the test machine is connected to the Internet.

UTD_OS

tests/user_test_data.bash.example/UTD_OS

Specify your operating system here, currently supported values:
debian|fedora|red hat|centos|ubuntu|opensuse|qubes dom0|other
Qubes VMs should be specified with their OS.

UTD_QUBES

tests/user_test_data.bash.example/UTD_QUBES

Specify whether you are running Qubes OS and in what environment here, possible values:
no|vm|dom0

UTD_QUBES_TESTVM

tests/user_test_data.bash.example/UTD_QUBES_TESTVM

If you're using Qubes OS, please specify a *disposable* virtual machine with a static name to be used for testing here. This test VM may crash, be destroyed or whatever - so please don't use a production VM!
Apart from that, the tests will be conducted with disposable VMs with dynamic names using your default template.

Example command to create such a test VM:

```
qvm-create --class DispVM --prop netvm='' --template nonet-dvm -l red d-testing
```

Can safely be ignored if you don't run Qubes OS-related tests.

UTD_QUBES_TESTVM_PERSISTENT

tests/user_test_data.bash.example/UTD_QUBES_TESTVM_PERSISTENT

If you're using Qubes OS, please specify a *persistent/non-disposable* virtual machine with a static name to be used for testing here. This test VM may crash, be destroyed or whatever - so please don't use a production VM!

Example command to create such a test VM:

```
qvm-create -l red --prop netvm='' testing-pers
```

Can safely be ignored if you don't run Qubes OS-related tests.

UTD_QUBES_DISPVM_TEMPLATE

tests/user_test_data.bash.example/UTD_QUBES_DISPVM_TEMPLATE

Qubes OS disposable VM template to use for testing. The test code will create disposable VMs from that template. It will *not* modify the template itself.
If no template is specified, the Qubes OS default template is used.

UTD_QUBES_DOM0_WD

tests/user_test_data.bash.example/UTD_QUBES_DOM0_WD

Full path to a folder in Qubes dom0 that can be created and deleted at will (working directory). Do **not** use an existing folder here!

Can safely be ignored if you don't run Qubes OS-related tests.

UTD_PW_FREE_USER

tests/user_test_data.bash.example/UTD_PW_FREE_USER

A user that allows password-less logons with sudo or su for testing.

If you run bats as root, neither sudo nor su should ask for a password.
Tests requiring admin privileges (e.g. mounting devices) may require this to be root or skip otherwise.

tests/ZZ_last.bats

This is not a real test, but can be used to execute some code *after* any bats tests are run.

0.1

Reference List

b_arr_contains

b_arr_getDeps

b_arr_join

b_arr_mapsAreEqual

b_blib_getDeps

B_CALLER_NAME

b_cdoc_cbPrintFirstParam

b_cdoc_cbPrintNewline

b_cdoc_generate

b_cdoc_generateBlibStyle

b_cdoc_getDeps

b_cdoc_getDocumentBeginCallback

b_cdoc_getDocumentEndCallback

b_cdoc_getExtractionRegex

b_cdoc_getFileFilterCallback

b_cdoc_getPostProcessingCallback

b_cdoc_getSpaceCallback

b_cdoc_setDocumentBeginCallback

b_cdoc_setDocumentEndCallback

b_cdoc_setExtractionRegex

b_cdoc_setFileFilterCallback

b_cdoc_setPostProcessingCallback

b_cdoc_setSpaceCallback

b_checkDeps

b_daemon_getDeps

b_daemon_getPid

B_DAEMON_ID

b_daemon_init

b_daemon_restart

b_daemon_start

b_daemon_status

b_daemon_statusPid

b_daemon_stop

b_date_addDays

b_date_diffSeconds

b_date_getDeps

b_defaultErrorHandler

b_dom0_attachFile

b_dom0_attachVMDisk

b_dom0_closeCrypt

b_dom0_copy

b_dom0_createLoopDeviceIfNecessary

b_dom0_crossAttachDevice

b_dom0_crossAttachFile

b_dom0_crossCopy

b_dom0_detachDevice

b_dom0_ensureRunning

b_dom0_enterEventLoop

b_dom0_execFuncIn

b_dom0_execIn

b_dom0_execStrIn

b_dom0_exists

b_dom0_getDeps

b_dom0_getDispVMs

b_dom0_getQvmBlockInfo

b_dom0_isMountedIn

b_dom0_isRunning

b_dom0_mountIfNecessary

b_dom0_openCrypt

b_dom0_parseQvmBlock

b_dom0_qvmRun

b_dom0_startDispVM

b_dom0_waitForFileIn

B_E

b_enforceUser

B_ERR

b_execFuncAs

b_flog__close

b_flog__defaultHeader

b_flogErrorHandler

b_flog__getDateFormat

b_flog__getDeps

b_flog__getHeaderFunction

b_flog__getLogReductionLinesLowerBound

b_flog__getLogReductionLinesUpperBound

b_flog__headerDateScriptSeverity

b_flog__headerDateSeverity

b_flog__init

b_flog__log

b_flog__printSeverity

b_flog__setDateFormat

b_flog__setHeaderFunction

b_flog__setLogReductionLinesApprox

b_flog__setLogReductionLinesLowerBound

b_flog__setLogReductionLinesUpperBound

B_FLOG_SEV

b_fs__createLoopDeviceIfNecessary

b_fs__getDeps

b_fs__getLastModifiedInDays

b_fs__getLineCount

b_fs__getMountpoints

b_fs__isEmptyDir

b_fs__mountIfNecessary

b_fs__parseSize

b_fs__waitForFile

b_generateStandalone

b_getBlibModules

b_getErrorHandler

B_HTTP_CHECKURLS

b_http_getDeps

b_http_getOnlineStatus

b_http_rawUrlDecode

b_http_rawUrlEncode

b_import

b_info

b_ini_get

b_ini_getBool

b_ini_getDeps

b_ini_getInt

b_ini_getString

b_ini_read

b_ipcm_change

b_ipcm_get

b_ipcm_getDeps

b_ipcm_getNamespace

b_ipcm_setNamespace

b_ipcm_unsetNamespace

b_ipcv_getDeps

b_ipcv_load

b_ipcv_loadNamespace

b_ipcv_save

b_ipcv_unset

b_ipcv_unsetNamespace

b_isFunction

b_isModule

B_LIB_DIR

b_listContains

b_mtx_create

b_mtx_forceRelease

b_mtx_getDeps

b_mtx_getSleepTime

b_mtx_pass

b_mtx_release

b_mtx_setSleepTime

b_mtx_try

b_mtx_waitFor

b_multiw_getDeps

b_multiw_getMaxHangTime

b_multiw_remove

b_multiw_setMaxHangTime

b_multiw_write

b_nop

b_osid_getDeps

b_osid_init

b_osid_isCentOS

b_osid_isDebian

b_osid_isDebianLike

b_osid_isFedora

b_osid_isFedoraLike

b_osid_isOpenSuse

b_osid_isQubesDom0

b_osid_isQubesVM

b_osid_isRedHat

b_osid_isUbuntu

b_printStackTrace

b_proc_getDeps

b_proc_pidExists

b_proc_waitForPid

B_RC

b_resetErrorHandler

B_SCRIPT

B_SCRIPT_DIR

B_SCRIPT_NAME

b_setBE

b_setErrorHandler

b_silence

b_str_getDeps

b_str_stripQuotes

b_str_trim

B_TCOLORS

b_tcolors_getDeps

B_TEST_MODE

b_traps_add

b_traps_getCodeFor

b_traps_getDeps

b_traps_prepend

b_traps_remove

B_TYPES_ENCODINGS

b_types_getDeps

b_types_isInteger

b_types_parseString

b_version

clearBlibTestState

endTimer

funcTimeout

loadBlib

loadBlibTestState

runSC

runSL

saveBlibState

skipIfNotQubesDom0

skipIfNoUserData

skipIfNoUserData

startTimer

testGetterSetter

TEST_STATE

UTD_ONLINE

UTD_OS

UTD_PW_FREE_USER

UTD_QUBES

UTD_QUBES_DISPVM_TEMPLATE

UTD_QUBES_DOM0_WD

UTD_QUBES_TESTVM

UTD_QUBES_TESTVM_PERSISTENT