

Bashmatic Usage Docs (v2.2.6)

Table of Contents

File lib/file.sh	4
file.temp()	4
file.normalize-files()	4
Example	4
file.ask.if-exists()	4
File lib/pids.sh	4
pids.stop-by-listen-tcp-ports()	4
Example	5
pid.stop-if-listening-on-port()	5
Example	5
File lib/bashit.sh	5
bashit-prompt-terraform()	5
bashit-install()	5
File lib/array.sh	5
array.has-element()	6
Example	6
array.includes()	6
array.join()	6
Example	6
Arguments	6
array.sort()	6
Example	6
array.sort-numeric()	7
Example	7
array.min()	7
Example	7
array.max()	7
Example	7
array.uniq()	7
Example	7
array.from.command()	8
Example	8
File lib/asciidoc.sh	8
asciidoc.rouge-themes()	8
File lib/output-utils.sh	8
is-dbg()	8
dbg()	8
File lib/audio.sh	8
lib/audio.sh	9
File lib/brew.sh	10
package.is-installed()	10
File lib/output.sh	11
section()	11
Arguments	11
File lib/video.sh	11
lib/video.sh	11

File lib/path.sh	12
path.strip-slash()	13
path.dirs()	13
Arguments	13
path.dirs.size()	13
path.dirs.uniq()	13
path.dirs.delete()	13
Arguments	14
path.uniq()	14
path.append()	14
path.prepend()	14
path.mutate.uniq()	14
path.mutate.delete()	14
path.mutate.append()	14
path.mutate.prepend()	14
PATH_add()	14
File lib/osx.sh	15
osx.app.is-installed()	15
Example	15
Arguments	15
Exit codes	15
File lib/db.sh	15
db.config.parse()	16
Example	16
db.psql.connect()	16
Example	16
db.psql.connect.just-data()	16
Example	16
db.psql.connect.table-settings-set()	16
Example	16
db.psql.db-settings()	17
Example	17
db.psql.connect.db-settings-pretty()	17
Example	17
Arguments	17
db.psql.connect.db-settings-toml()	17
Example	17
Arguments	17
db.actions.run-multiple()	17
Example	17
db.actions.pga()	18
File lib/shdoc.sh	18
lib/shdoc.sh	18
File lib/git.sh	18
git.cfgu()	18
Example	18
git.open()	18
Example	19
Arguments	19
File lib/package.sh	19
package.ensure.is-installed()	19
package.ensure.command-available()	19
Example	19

File lib/time.sh	19
time.with-duration.start()	19
Example	19
File lib/shasum.sh	20
shasum.set-command()	20
shasum.set-algo()	20
Example	20
shasum.sha()	20
shasum.sha-only()	21
shasum.sha-only-stdin()	21
shasum.to-hash()	21
Example	21
shasum.all-files()	21
Example	21
shasum.all-files-in-dir()	21
Example	21
File lib/pg.sh	22
pg.is-running()	22
pg.running.server-binaries()	22
pg.running.data-dirs()	22
pg.server-in-path.version()	22
File lib/dir.sh	22
dir.short-home()	22
File lib/config.sh	22
config.get-format()	23
config.set-file()	23
config.get-file()	23
config.dig()	23
Arguments	23
config.dig.pretty()	23
File lib/net.sh	23
net.is-host-port-protocol-open()	23
Arguments	23
File lib/is.sh	24
__is.validation.error()	24
Arguments	24
Exit codes	24
is-validations()	24
__is.validation.ignore-error()	24
__is.validation.report-error()	24
whenever()	24
Example	24
File lib/util.sh	25
util.rot13-stdin()	25
Example	25
File lib/pdf.sh	25
Bashmatic Utilities for PDF file handling	25
File bin/install-direnv	26
direnv.register()	26
File bin/regen-usage-docs	26
File bin/pdf-reduce	26
pdf.do.shrink()	26
File bin/scheck	26

manual-install()	26
Copyright & License	26

NOTICE: [shdoc](#) documentation is auto-extracted from the Bashmatic Sources.

File lib/file.sh

- [file.temp\(\)](#)
- [file.normalize-files\(\)](#)
- [file.ask.if-exists\(\)](#)

file.temp()

Creates a temporary file and returns it as STDOUT

file.normalize-files()

This function will rename all files passed to it as follows: spaces are replaced by dashes, non printable characters are removed, and the filename is lower cased.

Example

```
file.normalize-files "My Word Document.docx"
# my-word-document.docx
```

file.ask.if-exists()

Ask the user whether to overwrite the file

File lib/pids.sh

- [pids.stop-by-listen-tcp-ports\(\)](#)
- [pid.stop-if-listening-on-port\(\)](#)

pids.stop-by-listen-tcp-ports()

Finds any PID listening on one of the provided ports and stop them.

Example

```
pid.stop-by-listen-tcp-ports 4232 9578 "${PORT}"
```

pid.stop-if-listening-on-port()

Finds any PID listening the one port and an optional protocol (tcp/udp)

Example

```
pid.stop-if-listening-on-port 3000 tcp  
pid.stop-if-listening-on-port 8126 udp
```

File lib/bashit.sh

- [bashit-prompt-terraform\(\)](#)
- [bashit-install\(\)](#)

bashit-prompt-terraform()

Possible Bash It Powerline Prompt Modules

aws_profile battery clock command_number cwd dirstack gcloud go history_number hostname in_toolbox
in_vim k8s_context last_status node python_venv ruby scm shlvl terraform user_info wd

bashit-install()

Installs Bash-It Framework

File lib/array.sh

- [array.has-element\(\)](#)
- [array.includes\(\)](#)
- [array.join\(\)](#)
- [array.sort\(\)](#)
- [array.sort-numeric\(\)](#)
- [array.min\(\)](#)

- `array.max()`
- `array.uniq()`
- `array.from.command()`

`array.has-element()`

Returns "true" if the first argument is a member of the array passed as the second argument:

Example

```
$ declare -a array=("a string" test2000 moo)
if [[ $(array.has-element "a string" "${array[@]}") == "true" ]]; then
...
fi
```

`array.includes()`

Similar to `array.has-elements`, but does not print anything, just returns 0 if includes, 1 if not.

`array.join()`

Joins a given array with a custom string.

Example

```
$ declare -a array=(one two three)
$ array.join " " "${array[@]}"
$ array.join "-> " true "${array[@]}"
-> one
-> two
-> three
```

Arguments

- `@arg1`
- `@arg2`
- `@arg3 .`

`array.sort()`

Sorts the array alphanumerically and prints it to STDOUT

Example

```
declare -a unsorted=(hello begin again again)
local sorted="$(array.sort "${unsorted[@]}")"
```

array.sort-numeric()

Sorts the array numerically and prints it to STDOUT

Example

```
declare -a unsorted=(1 2 34 45 6)
local sorted="$(array.sort-numeric "${unsorted[@]}")"
```

array.min()

Returns a minimum integer from an array. Non-numeric elements are ignored and skipped over. Negative numbers are supported, but non-integers are not.

Example

```
$ declare -a array=(10 20 30 -5 5)
$ array.min "," "${array[@]}"
-5
```

array.max()

Returns a maximum integer from an array. Non-numeric elements are ignored and skipped over. Negative numbers are supported, but non-integers are not.

Example

```
$ declare -a array=(10 20 30 -5 5)
$ array.min "," "${array[@]}"
30
```

array.uniq()

Sorts and uniqs the array and prints it to STDOUT

Example

```
declare -a unsorted=(hello hello hello goodbye)
local uniqed="$(array.sort-numeric "${unsorted[@]}")"
```

array.from.command()

Creates an array variable, where each element is a line from a command output, which includes any spaces.

Example

```
array.from.command music_files "find . -type f -name '*.mp3'"  
echo "You have ${#music[@]} music files."
```

File lib/asciidoc.sh

Provides helper functions for dealing with asciidoc format.

- [asciidoc.rouge-themes\(\)](#)

asciidoc.rouge-themes()

Installs gem "rouge" and prints all available themes

File lib/output-utils.sh

- [is-dbg\(\)](#)
- [dbg\(\)](#)

is-dbg()

Checks if we have debug mode enabled

dbg()

Local debugging helper, activate it with DEBUG=1

File lib/audio.sh

lib/audio.sh

Audio conversions routines.

- `audio.file.frequency()`
- `audio.make.mp3s()`
- `audio.make.mp3()`
- `audio.file.mp3-to-wav()`
- `audio.dir.mp3-to-wav()`
- `.audio.karaoke.format()`
- `audio.dir.rename-wavs()`
- `audio.dir.rename-karaoke-wavs()`

audio.file.frequency()

Given a music audio file, determine its frequency.

audio.make.mp3s()

Given a folder of MP3 files, and an optional KHz specification, perform a sequential conversion from AIF/WAV format to MP3.

Example

```
audio.wav-to-mp3 [ file.wav | file.aif | file.aiff ] [ file.mp3 ]
```

audio.make.mp3()

Converts one AIF/WAV file to high-rez 320 Kbps MP3

audio.file.mp3-to-wav()

Decodes a folder with MP3 files back into WAV

audio.dir.mp3-to-wav()

assume a folder with a bunch of MP3s in subfolders

Example

```
same folder structure but under /Volumes/SDCARD.
```

`.audio.karaoke.format()`

Rename function for one filename to another. This particular function deals with files of this format:
Downloaded from karaoke-version.com:

Example

```
.audio.karaoke.format "Michael_Jackson_Billie_Jean(Drum_Backing_Track_(Drum_only))_248921.wav"  
=> michael_jackson_billie_jean--drum_backing_track-drum_only.wav
```

`audio.dir.rename-wavs()`

This function receives a format specification, and an optional directory as a second argument. Format specification is meant to map to a function `.audio.<format>.format` that's used as follows:
`.audio.<format>.format "file-name" => "new file name"`

Example

```
audio.dir.rename-wavs karaoke ~/Karaoke
```

`audio.dir.rename-karaoke-wavs()`

Renames wav files in the current folder (or the folder passed as an argument, based on the naming scheme downloaded from karaoke-version.com

Example

```
audio.dir.rename-karaoke-wavs "~/Karaoke"
```

File `lib/brew.sh`

- `package.is-installed()`

`package.is-installed()`

For each passed argument checks if it's installed.

File lib/output.sh

- `section()`

`section()`

Prints a "arrow-like" line using powerline characters

Arguments

- @arg1 Width (optional) — only interpreted as width if the first argument is a number.
 - @args Text to print
-

File lib/video.sh

lib/video.sh

Video conversions routines.

- `.video.install-deps()`
- `.video.convert.compress-shrinkwrap()`
- `.video.convert.compress-11()`
- `.video.convert.compress-12()`
- `.video.convert.compress-13()`
- `.video.convert.compress-21()`
- `.video.convert.compress-22()`
- `.video.convert.compress-23()`
- `.video.convert.compress-3()`
- `video.convert.compress()`

`.video.install-deps()`

Installs ffmpeg

`.video.convert.compress-shrinkwrap()`

Named after the author of a similar tool that does this:

.video.convert.compress-11()

Given two arguments (from), (to), performs a video recompression

.video.convert.compress-12()

Given two arguments (from), (to), performs a video recompression

.video.convert.compress-13()

Given two arguments (from), (to), performs a video recompression

.video.convert.compress-21()

Given two arguments (from), (to), performs a video recompression

.video.convert.compress-22()

Given two arguments (from), (to), performs a video recompression

.video.convert.compress-23()

Given two arguments (from), (to), performs a video recompression

.video.convert.compress-3()

Given two arguments (from), (to), performs a video recompression

video.convert.compress()

Given two arguments (from), (to), performs a video recompression according to the algorithm in the second argument.

Example

```
video.convert.compress bigfile.mov 13
```

File lib/path.sh

Utilities for managing the \$PATH variable

- `path.strip-slash()`
- `path.dirs()`
- `path.dirs.size()`
- `path.dirs.uniq()`
- `path.dirs.delete()`
- `path.uniq()`
- `path.append()`
- `path.prepend()`
- `path.mutate.uniq()`
- `path.mutate.delete()`
- `path.mutate.append()`
- `path.mutate.prepend()`
- `PATH_add()`

`path.strip-slash()`

Removes a trailing slash from an argument path

`path.dirs()`

Prints a new-line separated list of paths in PATH

Arguments

- `@arg1` A path to split, defaults to `$PATH`

`path.dirs.size()`

Prints the total number of paths in the path argument, which defaults to `$PATH`

`path.dirs.uniq()`

Prints all folders in `$PATH`, one per line, removing any duplicates, Does not mutate the `$PATH`

`path.dirs.delete()`

Deletes any number of folders from the PATH passed as the first string argument (defaults to `$PATH`). Does not mutate the `$PATH`, just prints the result to STDOUT

Arguments

- @arg1 String representation of a PATH, eg `"/bin:/usr/bin:/usr/local/bin"`
- @arg2 An array of paths to be removed from the PATH

`path.uniq()`

Removes duplicates from the \$PATH (or argument) and prints the results in the PATH format (column-joined). DOES NOT mutate the actual \$PATH

`path.append()`

Appends a new directory to the \$PATH and prints the result to STDOUT, Does NOT mutate the actual \$PATH

`path.prepend()`

Prepends a new directory to the \$PATH and prints to STDOUT, If one of the arguments already in the PATH its moved to the front. DOES NOT mutate the actual \$PATH

`path.mutate.uniq()`

Removes any duplicates from \$PATH and exports it.

`path.mutate.delete()`

Deletes paths from the PATH provided on the command line

`path.mutate.append()`

Appends valid directories to those in the PATH, and exports the new value of the PATH

`path.mutate.prepend()`

Prepends valid directories to those in the PATH, and exports the new value of the PATH

`PATH_add()`

This function exists within direnv, but since we are sourcing in `.envrc` we need to have this defined to avoid errors.

File lib/osx.sh

OSX Specific Helpers and Utilities

- [osx.app.is-installed\(\)](#)

osx.app.is-installed()

Checks if a given parameter matches any of the installed applications under /Applications and ~/Applications

By the default prints the matched application. Pass -q as a second argument to disable output.

Example

```
> osx.app.is-installed safari
Safari.app
> osx.app.is-installed safari -q && echo installed
installed
> osx.app.is-installed microsoft -c
6
```

Arguments

- \$1 (a): string value to match (case insentively) for an app name
- \$2.. additional arguments to the last invocation of grep

Exit codes

- 0: if match was found
- 1: if not

File lib/db.sh

- [db.config.parse\(\)](#)
- [db.psql.connect\(\)](#)
- [db.psql.connect.just-data\(\)](#)
- [db.psql.connect.table-settings-set\(\)](#)
- [db.psql.db-settings\(\)](#)
- [db.psql.connect.db-settings-pretty\(\)](#)
- [db.psql.connect.db-settings-toml\(\)](#)
- [db.actions.run-multiple\(\)](#)

- `db.actions.pga()`

`db.config.parse()`

Returns a space-separated values of db host, db name, username and password

Example

```
db.config.set-file ~/.db/database.yml
db.config.parse development
#=> hostname dbname dbuser dbpass
declare -a params=($(db.config.parse development))
echo ${params[0]} # host
```

`db.psql.connect()`

Connect to one of the databases named in the YAML file, and optionally pass additional arguments to psql. Informational messages are sent to STDERR.

Example

```
db.psql.connect production
db.psql.connect production -c 'show all'
```

`db.psql.connect.just-data()`

Similar to the `db.psql.connect`, but outputs just the raw data with no headers.

Example

```
db.psql.connect.just-data production -c 'select datname from pg_database;'
```

`db.psql.connect.table-settings-set()`

Set per-table settings, such as autovacuum, eg:

Example

```
db.psql.connect.table-settings-set prod users autovacuum_analyze_threshold 1000000
db.psql.connect.table-settings-set prod users autovacuum_analyze_scale_factor 0
```


db.psql.db-settings()

Print out PostgreSQL settings for a connection specified by args

Example

```
db.psql.db-settings -h localhost -U postgres appdb
```

db.psql.connect.db-settings-pretty()

Print out PostgreSQL settings for a named connection

Example

```
db.psql.connect.db-settings-pretty primary
```

Arguments

- @arg1 dbname database entry name in ~/.db/database.yml

db.psql.connect.db-settings-toml()

Print out PostgreSQL settings for a named connection using TOML/ini format.

Example

```
db.psql.connect.db-settings-toml primary > primary.ini
```

Arguments

- @arg1 dbname database entry name in ~/.db/database.yml

db.actions.run-multiple()

Executes multiple commands by passing them to psql each with -c flag. This allows, for instance, setting session values, and running commands such as VACUUM which can not run within an implicit transaction started when joining multiple statements with ";"

Example

```
$ db -q run my_database 'set default_statistics_target to 10; show default_statistics_target; vacuum users'  
ERROR: VACUUM cannot run inside a transaction block
```

db.actions.pga()

Installs (if needed) pg_activity and starts it up against the connection

File lib/shdoc.sh

lib/shdoc.sh

Helpers to install gawk and shdoc properly.

see `${BASHMATIC_HOME}/lib/shdoc.md` for an example of how to use SHDOC. and also [project's github page](#).

- [gawk.install\(\)](#)

gawk.install()

Installs gawk into `/usr/local/bin/gawk`

File lib/git.sh

- [git.cfgu\(\)](#)
- [git.open\(\)](#)

git.cfgu()

Sets or gets user values from global gitconfig.

Example

```
git.cfgu email  
git.cfgu email kigster@gmail.com  
git.cfgu
```

git.open()

Reads the remote of a repo by name provided as an argument (or defaults to "origin") and opens it in the browser.

Example

```
git clone git@github.com:kigster/bashmatic.git
cd bashmatic
source init.sh
git.open
git.open origin # same thing
```

Arguments

- **\$1** (optional): name of the remote to open, defaults to "origin"
-

File lib/package.sh

- `package.ensure.is-installed()`
- `package.ensure.command-available()`

`package.ensure.is-installed()`

fr

`package.ensure.command-available()`

Example

```
In this example we skip installation if 'gem' exists and in the PATH.
Otherwise we install the package and retry, and return if not found
```

File lib/time.sh

- `time.with-duration.start()`

`time.with-duration.start()`

Starts a time for a given name space

Example

```
time.with-duration.start moofie
# ... time passes
time.with-duration.end moofie 'Moofie is now this old: '
# ... time passes
time.with-duration.end moofie 'Moofie is now very old: '
time.with-duration.clear moofie
```

File lib/shasum.sh

SHA Functions

SHASUM related functions, that compute SHA for a single file, collection of files, or entire directories.

- [shasum.set-command\(\)](#)
- [shasum.set-algo\(\)](#)
- [shasum.sha\(\)](#)
- [shasum.sha-only\(\)](#)
- [shasum.sha-only-stdin\(\)](#)
- [shasum.to-hash\(\)](#)
- [shasum.all-files\(\)](#)
- [shasum.all-files-in-dir\(\)](#)

shasum.set-command()

Override the default SHA command and algorithm Default is shasum -a 256

shasum.set-algo()

Override the default SHA algorithm

Example

```
$ shasum.set-algo 256
```

shasum.sha()

Compute SHA for all given files, ignore STDERR NOTE: first few arguments will be passed to the shasum command, or whatever you set via shasum.set-command.

shasum.sha-only()

Print SHA ONLY removing the file components

shasum.sha-only-stdin()

Print SHA ONLY removing the file components

shasum.to-hash()

This function populates a pre-declare associative array with filenames mapped to their SHAs, but only in the current directory Call dbg-on to enable additional debugging info.

Example

```
$ declare -A file_shas
$ shasum.to-hash file_shas $(find . -type f -maxdepth 2)
$ echo "Total of ${#file_shas[@]} files in the hash"
```

shasum.all-files()

For a given array of files, sort them, take a SHA of each file, and return a single SHA finger-printing this set of files. # NOTE: the files are sorted prior to hashing, so the return SHA should ONLY change when files are either changed, or added/removed. Only computes SHA of the files provided, does not recurse into folders

Example

```
$ shasum.all-files *.cpp
```

shasum.all-files-in-dir()

For a given directory and an optional file pattern, use find to grab every single file (that matches optional pattern) and return a single SHA

Example

```
$ shasum.all-files-in-dir . '*.pdf'
cc35aad389e61942c75e111f1eddb634d74b4b1
```

File lib/pg.sh

- [pg.is-running\(\)](#)
- [pg.running.server-binaries\(\)](#)
- [pg.running.data-dirs\(\)](#)
- [pg.server-in-path.version\(\)](#)

pg.is-running()

Returns true if PostgreSQL is running locally

pg.running.server-binaries()

if one or more PostgreSQL instances is running locally, prints each server's binary postgres file path

pg.running.data-dirs()

For each running server prints the data directory

pg.server-in-path.version()

Grab the version from postgres binary in the PATH and remove fractional sub-version

File lib/dir.sh

- [dir.short-home\(\)](#)

dir.short-home()

Replaces the first part of the directory that matches \${HOME} with '~/'

File lib/config.sh

- [config.get-format\(\)](#)
- [config.set-file\(\)](#)
- [config.get-file\(\)](#)
- [config.dig\(\)](#)

- [config.dig.pretty\(\)](#)

config.get-format()

Get current format

config.set-file()

Set the default config file

config.get-file()

Get the file name

config.dig()

Reads the value from a two-level configuration hash

Arguments

- @arg1 hash key
- @arg2 hash sub-key

config.dig.pretty()

Uses jq utility to format JSON with color, supports partial

File lib/net.sh

- [net.is-host-port-protocol-open\(\)](#)

net.is-host-port-protocol-open()

Uses pingless connection to check if a remote port is open Requires sudo for UDP

Arguments

- @arg1 host
 - @arg2 port
 - @arg3 [optional] protocol (defaults to "tcp", supports also "udp")
-

File lib/is.sh

Various validations and asserts that can be chained and be explicit in a DSL-like way.

- `<<isvalidationerror,is.validation.error(>>`
- `is-validations()`
- `<<isvalidationignore-error,is.validation.ignore-error(>>`
- `<<isvalidationreport-error,is.validation.report-error(>>`
- `whenever()`

`__is.validation.error()`

Invoke a validation on the value, and process the invalid case using a customizable error handler.

Arguments

- `@arg1 func` Validation function name to invoke
- `@arg2 var` Value under the test
- `@arg4 error_func` Error function to call when validation fails

Exit codes

- `0`: if validation passes

`is-validations()`

Returns the list of validation functions available

`__is.validation.ignore-error()`

Private function that ignores errors

`__is.validation.report-error()`

Private function that ignores errors

`whenever()`

a convenient DSL for validating things

Example


```
whenever /var/log/postgresql.log is.an-empty-file && {  
    touch /var/log/postgresql.log  
}
```

File lib/util.sh

Miscellaneous utilities.

- [util.rot13-stdin\(\)](#)

util.rot13-stdin()

Convert STDIN using rot13

Example

```
echo "test" | util.rot13-stdin
```

File lib/pdf.sh

Bashmatic Utilities for PDF file handling

Install and uses GhostScript to manipulate PDFs.

- [pdf.combine\(\)](#)

pdf.combine()

Combine multiple PDFs into a single one using ghostscript.

Example

```
pdf.combine ~/merged.pdf 'my-book-chapter*'
```

Arguments

- **\$1** (pathname): to the merged file

- ... (the): rest of the PDF files to combine
-

File bin/install-direnv

Add direnv hook to shell RC files

- [direnv.register\(\)](#)

direnv.register()

Add direnv hook to shell RC files

File bin/regen-usage-docs

Regenerates USAGE.adoc && USAGE.pdf

File bin/pdf-reduce

- [pdf.do.shrink\(\)](#)

pdf.do.shrink()

shrinks PDF

File bin/scheck

- [manual-install\(\)](#)

manual-install()

Manually Download and Install ShellCheck

Copyright & License

- Copyright © 2017-2021 Konstantin Gredeskoul, All rights reserved.
-

- Distributed under the MIT License.