

# Bashmatic Usage Docs (v3.1.3)

## Table of Contents

File <code>lib/yarn.sh</code> .....	7
<code>yarn_install()</code> .....	7
<code>yarn_sha()</code> .....	8
File <code>lib/dropbox.sh</code> .....	8
<code>function dropbox.ignore {</code> .....	8
<code>See also</code> .....	8
<code>dropbox.unignore()</code> .....	8
<code>See also</code> .....	8
File <code>lib/file.sh</code> .....	8
<code>file.temp()</code> .....	9
<code>file.normalize-files()</code> .....	9
<code>Example</code> .....	9
<code>file.first-is-newer-than-second()</code> .....	9
<code>file.ask.if-exists()</code> .....	9
<code>file.install-with-backup()</code> .....	9
<code>Example</code> .....	9
<code>Arguments</code> .....	9
<code>file.last-modified-date()</code> .....	10
<code>file.last-modified-year()</code> .....	10
<code>file.last-modified-millis()</code> .....	10
<code>file.size()</code> .....	10
<code>file.size.mb()</code> .....	10
<code>file.size.gb()</code> .....	10
<code>file.list.filter-existing()</code> .....	10
<code>file.list.filter-non-empty()</code> .....	10
<code>file.count.lines()</code> .....	10
<code>file.count.words()</code> .....	10
<code>file.find()</code> .....	10
<code>dir.find()</code> .....	11
<code>ls.mb()</code> .....	11
<code>ls.gb()</code> .....	11
File <code>lib/url.sh</code> .....	11
<code>url.cert.is-valid()</code> .....	11
<code>Arguments</code> .....	11
<code>url.cert.domain()</code> .....	11
<code>Example</code> .....	11

url.host.is-valid()	11
url.cert.info()	12
File lib/pids.sh	12
pids.stop-by-listen-tcp-ports()	12
Example	12
pid.stop-if-listening-on-port()	12
Example	12
File lib/bashit.sh	12
bashit-prompt-terraform()	12
bashit-install()	12
File lib/array.sh	13
array.has-element()	13
Example	13
array.includes()	13
array.join()	13
Example	13
Arguments	14
array.sort()	14
Example	14
array.sort-numeric()	14
Example	14
array.min()	14
Example	14
array.force-range()	14
Example	14
array.max()	15
Example	15
array.uniq()	15
Example	15
array.from.command()	15
Example	15
File lib/asciidoc.sh	15
asciidoc.rouge-themes()	16
File lib/output-utils.sh	16
is-dbg()	16
dbg()	16
File lib/audio.sh	16
lib/audio.sh	16
File lib/brew.sh	18
package.is-installed()	18
File lib/output.sh	18

output.screen-width.actual()	18
output.screen-height.actual()	18
section()	18
Arguments	18
File lib/usage.sh	18
usage-widget()	19
Example	19
File lib/file-helpers.sh	20
.file.make_executable()	20
File lib/video.sh	20
lib/video.sh	20
File lib/path.sh	22
path.strip-slash()	22
path.dirs()	22
Arguments	22
path.dirs.size()	22
path.dirs.uniq()	22
path.dirs.delete()	23
Arguments	23
path.uniq()	23
PATH.uniqify()	23
path.append()	23
path.prepend()	23
path.mutate.uniq()	23
path.mutate.delete()	23
path.mutate.append()	23
path.mutate.prepend()	23
PATH_add()	23
path.absolute()	24
File lib/osx.sh	24
osx.app.is-installed()	24
Example	24
Arguments	24
Exit codes	24
osx.detect-cpu()	24
Example	25
File lib/bashmatic.sh	25
bashmatic.is-developer()	25
File lib/db.sh	25
db.config.parse()	25
Example	25

<code>db.psql.connect()</code>	26
Example	26
<code>db.psql.connect.just-data()</code>	26
Example	26
<code>db.psql.connect.table-settings-set()</code>	26
Example	26
<code>db.psql.db-settings()</code>	26
Example	26
<code>db.psql.connect.db-settings-pretty()</code>	26
Example	26
Arguments	27
<code>db.psql.connect.db-settings-toml()</code>	27
Example	27
Arguments	27
<code>db.actions.run-multiple()</code>	27
Example	27
<code>db.actions.pga()</code>	27
File <code>lib/shdoc.sh</code>	27
<code>lib/shdoc.sh</code>	27
File <code>lib/git.sh</code>	28
<code>git.cfgu()</code>	28
Example	28
<code>git.repo()</code>	28
Example	28
Arguments	28
<code>git.repo.current()</code>	28
<code>git.cfg.get()</code>	29
Example	29
Arguments	29
File <code>lib/package.sh</code>	29
<code>package.ensure.is-installed()</code>	29
<code>package.ensure.command-available()</code>	29
Example	29
File <code>lib/time.sh</code>	29
<code>date.now.with-time()</code>	30
Example	30
<code>time.with-duration.start()</code>	30
Example	30
<code>time.with-duration()</code>	30
Example	30
Arguments	30

time.a-command()	30
Arguments	31
File lib/shasum.sh	31
shasum.set-command()	31
shasum.set-algo()	31
Example	31
shasum.sha()	31
shasum.sha-only()	31
shasum.sha-only-stdin()	32
shasum.to-hash()	32
Example	32
shasum.all-files()	32
Example	32
shasum.all-files-in-dir()	32
Example	32
sha()	32
File lib/runtime-config.sh	32
is.dry-run.on()	33
Example	33
is.dry-run.off()	33
Example	33
set.dry-run.on()	33
Example	33
set.dry-run.off()	33
Example	33
File lib/color.sh	34
color.current-background()	34
File lib/pg.sh	34
pg.is-running()	34
pg.running.server-binaries()	34
pg.running.data-dirs()	34
pg.server-in-path.version()	34
File lib/7z.sh	34
lib/7z.sh	34
File lib/dir.sh	35
dir.with-file()	35
Arguments	35
dir.short-home()	35
File lib/config.sh	35
config.get-format()	35
config.set-file()	35

config.get-file()	35
config.dig()	35
Arguments	36
config.dig.pretty()	36
File lib/flatten.sh	36
flatten-file()	36
Example	36
File lib/nvm.sh	36
nvm.is-valid-dir()	36
nvm.detect()	37
nvm.install()	37
nvm.load()	37
File lib/net.sh	37
net.is-host-port-protocol-open()	37
Arguments	37
File lib/is.sh	37
__is.validation.error()	38
Arguments	38
Exit codes	38
is-validations()	38
__is.validation.ignore-error()	38
__is.validation.report-error()	39
is.not-blank()	39
is.blank()	39
is.empty()	39
is.not-a-blank-var()	39
is.a-non-empty-file()	39
is.an-empty-file()	39
is.a-directory()	39
is.an-existing-file()	39
is.a-function.invoke()	39
is.a-function()	39
is.a-variable()	39
is.a-non-empty-array()	40
is.sourced-in()	40
is.a-script()	40
is.integer()	40
See also	40
is.an-integer()	40
is.numeric()	40
is.command()	40

is.a-command()	40
is.missing()	40
is.alias()	40
is.zero()	40
is.non.zero()	41
whenever()	41
Example	41
File lib/util.sh	41
system.uname()	41
util.random-number()	41
util.generate-password()	41
util.random-string.of-length()	41
File lib/runtime.sh	41
run.print-variables()	42
run.inspect-vars()	42
File lib/pdf.sh	42
Bashmatic Utilities for PDF file handling	42
File bin/install-direnv	42
direnv.register()	43
File bin/regen-usage-docs	43
File bin/pdf-reduce	43
pdf.do.shrink()	43
File bin/scheck	43
manual-install()	43
Copyright & License	43

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

## File lib/yarn.sh

- [yarn\\_install\(\)](#)
- [yarn\\_sha\(\)](#)

### yarn\_install()

Installs YARN via npm if not found; then runs yarn install Note that yarn install is skipped if package.json and yarn.lock haven't changed since the last run of yarn install.

## yarn\_sha()

Prints to STDOUT the SHA based on package.json and yarn.lock

---

## File `lib/dropbox.sh`

- `function dropbox.ignore {`
- `dropbox.unignore()`

### `function dropbox.ignore {`

Set file to be ignored by Dropbox

#### See also

- <https://help.dropbox.com/files-folders/restore-delete/ignored-files>

## `dropbox.unignore()`

Set a file or directorhy to be ignored by Dropbox

#### See also

- <https://help.dropbox.com/files-folders/restore-delete/ignored-files>
- 

## File `lib/file.sh`

- `file.temp()`
  - `file.normalize-files()`
  - `file.first-is-newer-than-second()`
  - `file.ask.if-exists()`
  - `file.install-with-backup()`
  - `file.last-modified-date()`
  - `file.last-modified-year()`
  - `file.last-modified-millis()`
  - `file.size()`
  - `file.size.mb()`
  - `file.size.gb()`
  - `file.list.filter-existing()`
-



- `file.list.filter-non-empty()`
- `file.count.lines()`
- `file.count.words()`
- `file.find()`
- `dir.find()`
- `ls.mb()`
- `ls.gb()`

## `file.temp()`

Creates a temporary file and returns it as STDOUT shellcheck disable=SC2120

## `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.first-is-newer-than-second()`

A super verbose shortcut to `[[ file -nt file2 ]]`

## `file.ask.if-exists()`

Ask the user whether to overwrite the file

## `file.install-with-backup()`

Installs a given file into a provided destination, while making a backup of the destination if it already exists.

### Example

```
file.install-with-backup conf/.psqlrc ~/.psqlrc backup-strategy-function
```

### Arguments

- `@arg1` File to backup
- `@arg2` Destination

- @arg3 [optional] Shortname of the optional backup strategy: 'bak' or 'folder'.

## **file.last-modified-date()**

Prints the file's last modified date

## **file.last-modified-year()**

Prints the year of the file's last modified date

## **file.last-modified-millis()**

Prints the file's last modified date expressed as milliseconds

## **file.size()**

Returns the file size in bytes

## **file.size.mb()**

Prints the file size expressed in Mb (and up to 1 decimal point)

## **file.size.gb()**

Prints the file size expressed in Gb (and up to 1 decimal point)

## **file.list.filter-existing()**

For each argument prints only those that represent existing files

## **file.list.filter-non-empty()**

For each argument prints only those that represent non-empty files

## **file.count.lines()**

Prints the number of lines in the file

## **file.count.words()**

Prints the number of lines in the file

## **file.find()**

Invokes UNIX find command searching for files (not folders) matching the first argument in the name.

## dir.find()

Invokes UNIX find command searching for folders (not files) matching the first argument in the name.

## ls.mb()

Prints all folders sorted by size, and size printed in Mb

## ls.gb()

Prints all folders sorted by size, and size printed in Gb

---

## File lib/url.sh

- [url.cert.is-valid\(\)](#)
- [url.cert.domain\(\)](#)
- [url.host.is-valid\(\)](#)
- [url.cert.info\(\)](#)

## url.cert.is-valid()

Returns 0 if the certificate is valid of the domain passed as an argument.

### Arguments

- @arg0 domain or a complete https url

## url.cert.domain()

Prints the common name for which the SSL certificate is registered

### Example

```
❯ url.cert.domain google.com
*.google.com
```

## url.host.is-valid()

Returns 0 when the argument is a valid Internet host resolvable via DNS. Otherwise returns 255 and prints an error to STDERR.

## url.cert.info()

Returns the SSL information about the remote certificate

---

## 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

```
pids.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.force-range()`
- `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.force-range()

Given a numeric argument, and an additional array of numbers, determines the min/max range of the array and prints out the number if it's within the range of array's min and max. Otherwise prints out either min or max.

### Example

```
$ array.force-range 26 0 100
```

```
# => 26
$ array.force-range 26 60 100
# => 60
```

## 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 `export BASHMATIC_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](http://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

## 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`

- `output.screen-width.actual()`
- `output.screen-height.actual()`
- `section()`

### `output.screen-width.actual()`

OS-independent way to determine screen width.

### `output.screen-height.actual()`

OS-independent way to determine screen height.

### `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/usage.sh`

- `usage-widget()`

## usage-widget()

This is a massive hack and I am ashamed to have written it. With that out of the way, here we go. This command generates a pretty usage box for a tool or another command.

### Example

```
usage-widget [-]<width> \           # box width. If it starts with "-"
forces cache wipe.                  #
    "command [flags] <arg1 ... >" \   # <-- USAGE
    "This command is beyond description." \ # <-- DESCRIPTION
    "[@]string" \                     # <-- This and subsequent lines may
optionally start with "@" symbol,    #
    "[@]string" \                     #      which will turn them into sub-
headings:                             #
    "[@]string" \
    "[@]string"
usage-widget 90 \
    "command [flags] <arg1 ... >" \
    "This command is beyond description." \
    "@examples" \
    "Some examples will follow" \
    "And others won't."
```

```

┌
├  USAGE:          command [flags] <arg1 ... >
└
```

```

┌
├  DESCRIPTION:    This command is beyond description.
└
```

```

┌
├  EXAMPLES:
└
```

```

    Some examples will follow
```

```

    And others won't.
```

## File `lib/file-helpers.sh`

- `.file.make_executable()`

### `.file.make_executable()`

Makes a file executable but only if it already contains a "bang" line at the top.

---

## File `lib/video.sh`

# `lib/video.sh`

Video conversions routines.

- `.destination-file-name()`
- `.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.filename.encoded()`
- `video.install.dependencies()`
- `video.encode()`
- `video.squeeze()`

### `.destination-file-name()`

Generate a destination file name for the compressed items.

### `.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.filename.encoded()**

Given the source file passed as an argument, and the name of the encoding algorithm, prints the name of the destination file (which will be lower-cased, no spaces, and contain the algorithm)

## **video.install.dependencies()**

Installs ffmpeg and other dependencies

## **video.encode()**

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

### **Example**

```
video.encode bigfile.mov 13 smallerfile.mkv
@arg1 File to convert
@arg2 Name of the algorithm, defaults to 11
@arg3 Optional output file
```

## **video.squeeze()**

# 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.uniqify()`
- `path.append()`
- `path.prepend()`
- `path.mutate.uniq()`
- `path.mutate.delete()`
- `path.mutate.append()`
- `path.mutate.prepend()`
- `PATH_add()`
- `path.absolute()`

## 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.uniqify()

Using sed and tr uniq the PATH without re-sorting it.

## 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.

## path.absolute()

Returns an absolute version of a given path

---

## File lib/osx.sh

OSX Specific Helpers and Utilities

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

## 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

## osx.detect-cpu()

This function checks the architecture of the CPU, but also is able to detect when M1 system is running under Rosetta.



## Example

```
local -a ostype=( $(osx.detect-cpu) )
local cpu=${ostype[0]}
local emulation="${ostype[1]}"
```

## File `lib/bashmatic.sh`

- `bashmatic.is-developer()`

### `bashmatic.is-developer()`

True if `.envrc.local` file is present. We take it as a sign you may be developing bashmatic.

## 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.0

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.repo()`
- `git.repo.current()`
- `git.cfg.get()`

## git.cfgu()

Sets or gets user values from global gitconfig.

### Example

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

## git.repo()

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"

## git.repo.current()

Returns a URL on Github website that points to the . README on the current branch.

## git.cfg.get()

Prints the value from github config

### Example

```
git.cfg.get github.token user.name user.email  
dsf09098f09ds8f0s98df09809  
John Doe  
jonny@hotmail.com
```

### Arguments

- @arg1 [ local | global ] which config to look at (defaults to global)
- @arg2... tokens to print

## 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

- [date.now.with-time\(\)](#)
- [time.with-duration.start\(\)](#)
- [time.with-duration\(\)](#)
- [time.a-command\(\)](#)

## date.now.with-time()

Prints the complete date with time up to milliseconds

### Example

```
2022-05-03 14:29:52.302
```

## 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
```

## time.with-duration()

Runs the given command and prints the time it took

### Example

```
time.with-duration quiet "{ sleep 1; ls -al; sleep 2; date; sleep 1; }"
time.with-duration quiet verbose "{ sleep 1; ls -al; sleep 2; date; sleep 1; }"
```

### Arguments

- @arg1 [quiet] to silence command output
- @arg2 [verbose] to print the command before running the
- @arg3 [secret] do not print the command before running it (in case sensitive)

## time.a-command()

This function receives a command to execute as an argument. The command is executed as 'eval "\$@"'; meanwhile the start/end times are measured, and the following string is printed at the end: eg. "4 minutes 24.345 seconds"

## Arguments

- @args Command to run
- 

# 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()`
- `sha()`

## `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
```

## sha()

sha256

---

## File `lib/runtime-config.sh`

- [is.dry-run.on\(\)](#)



- `is.dry-run.off()`
- `set.dry-run.on()`
- `set.dry-run.off()`

## `is.dry-run.on()`

Returns 0 when dry-run flag was set, 1 otherwise.

### Example

```
set.dry-run.on  
is.dry-run.on || rm -f ${temp}
```

## `is.dry-run.off()`

Returns 0 when dry-run flag was set, 1 otherwise.

### Example

```
set.dry-run.off  
is.dry-run.on || rm -f ${temp}
```

## `set.dry-run.on()`

Returns 0 when dry-run flag was set, 1 otherwise.

### Example

```
set.dry-run.on  
is.dry-run.on || run "ls -al"
```

## `set.dry-run.off()`

Returns 1 when dry-run flag was set, 0 otherwise.

### Example

```
set.dry-run.on  
is.dry-run.on || run "ls -al"
```

## File `lib/color.sh`

- `color.current-background()`

### `color.current-background()`

Prints the background color of the terminal, assuming terminal responds to the escape sequence. More info: <https://stackoverflow.com/questions/2507337/how-to-determine-a-terminals-background-color>

---

## 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/7z.sh`

## `lib/7z.sh`

p7zip conversions routines.

---

## File lib/dir.sh

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

### dir.with-file()

Returns the first folder above the given that contains a file.

#### Arguments

- @arg1 file without the path to search for, eg ".evnrc"
- @arg2 Starting file path to search

### 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/flatten.sh`

- `flatten-file()`

## flatten-file()

Given a long path to a file, possibly with spaces in cluded and a desintation as a second argument, generates a flat pathname and copies the first argument there.

### Example

```
❯ tree -Q "33 Retro Synth/"
"33 Retro Synth/"
├── "001 Retro Synth - A Synth Primer.en.srt"
├── "001 Retro Synth - A Synth Primer.mp4"
├── "002 Retro Synth - Oscillator.en.srt"
└── "002 Retro Synth - Oscillator.mp4"
❯
  flatten-file "33 Retro Synth/001 Retro Synth - A Synth Primer.mp4"
@arg1 -n | --dry-run (optional)
@arg2 source path
@arg3 dest paths
```

## File `lib/nvm.sh`

- `nvm.is-valid-dir()`
- `nvm.detect()`
- `nvm.install()`
- `nvm.load()`

## nvm.is-valid-dir()

Returns true if `NVM_DIR` is correctly set, OR if a directory passed as an argument contains `nvm.sh`

---

## **nvm.detect()**

Returns success and exports NVM\_DIR whenever nvm.sh is found underneath any of the possible locations tried.

## **nvm.install()**

Installs NVM via Curl if not already installed.

## **nvm.load()**

Load

---

## **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()>>`
- [is.not-blank\(\)](#)
- [is.blank\(\)](#)
- [is.empty\(\)](#)
- [is.not-a-blank-var\(\)](#)
- [is.a-non-empty-file\(\)](#)
- [is.an-empty-file\(\)](#)

- `is.a-directory()`
- `is.an-existing-file()`
- `is.a-function.invoke()`
- `is.a-function()`
- `is.a-variable()`
- `is.a-non-empty-array()`
- `is.sourced-in()`
- `is.a-script()`
- `is.integer()`
- `is.an-integer()`
- `is.numeric()`
- `is.command()`
- `is.a-command()`
- `is.missing()`
- `is.alias()`
- `is.zero()`
- `is.non.zero()`
- `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

## **is.not-blank()**

is.not-blank <arg></arg>

## **is.blank()**

is.blank <arg></arg>

## **is.empty()**

is.empty <arg></arg>

## **is.not-a-blank-var()**

is.not-a-blank-var <var-name></var-name>

## **is.a-non-empty-file()**

is.a-non-empty-file <file></file>

## **is.an-empty-file()**

is.an-empty-file <file></file>

## **is.a-directory()**

is.a-directory <dir></dir>

## **is.an-existing-file()**

is.an-existing-file <file></file>

## **is.a-function.invoke()**

if the argument passed is a value function, invoke it

## **is.a-function()**

verifies that the argument is a valid shell function

## **is.a-variable()**

verifies that the argument is a valid and defined variable

## **is.a-non-empty-array()**

verifies that the argument is a non-empty array

## **is.sourced-in()**

verifies that the argument is a valid and defined variable

## **is.a-script()**

returns success if the current script is executing in a subshell

## **is.integer()**

returns success if the argument is an integer

### **See also**

- <https://stackoverflow.com/questions/806906/how-do-i-test-if-a-variable-is-a-number-in-bash>

## **is.an-integer()**

returns success if the argument is an integer

## **is.numeric()**

returns success if the argument is numeric, eg. float

## **is.command()**

returns success if the argument is a valid command found in the \$PATH

## **is.a-command()**

returns success if the argument is a valid command found in the \$PATH

## **is.missing()**

returns success if the command passed as an argument is not in \$PATH

## **is.alias()**

returns success if the argument is a current alias

## **is.zero()**

returns success if the argument is a numerical zero



## is.non.zero()

returns success if the argument is not a zero

## 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.

- [system.uname\(\)](#)
- [util.random-number\(\)](#)
- [util.generate-password\(\)](#)
- [util.random-string.of-length\(\)](#)

## system.uname()

Finds the exact absolute path of the [uname](#) utility on a unix file system.

## util.random-number()

Generates a random number up to 1000000

## util.generate-password()

Generates a password of a given length

## util.random-string.of-length()

Generates a random string of a given length

---

## File lib/runtime.sh

- [run.print-variables\(\)](#)

- [run.inspect-vars\(\)](#)

## **run.print-variables()**

Adds a variable to the list of the variables to be obfuscated

## **run.inspect-vars()**

Prints values of all variables starting with prefixes in args

---

## **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 © 2017-2022 Konstantin Gredeksoul, All rights reserved.
- Distributed under the MIT License.