

# Bashmatic Usage Docs (v2.5.0)

## Table of Contents

File lib/file.sh	5
file.temp()	5
file.normalize-files()	5
Example	6
file.first-is-newer-than-second()	6
file.ask-if-exists()	6
file.install-with-backup()	6
Example	6
Arguments	6
File lib/pids.sh	6
pids.stop-by-listen-tcp-ports()	6
Example	6
pid.stop-if-listening-on-port()	7
Example	7
File lib/bashit.sh	7
bashit-prompt-terraform()	7
bashit-install()	7
File lib/array.sh	7
array.has-element()	8
Example	8
array.includes()	8
array.join()	8
Example	8
Arguments	8
array.sort()	8
Example	8
array.sort-numeric()	9
Example	9
array.min()	9
Example	9
array.force-range()	9
Example	9
array.max()	9
Example	9
array.uniq()	10
Example	10

array.from.command()	10
Example	10
File lib/asciidoc.sh	10
asciidoc.rouge-themes()	10
File lib/output-utils.sh	10
is-dbg()	10
dbg()	10
File lib/audio.sh	11
lib/audio.sh	11
File lib/brew.sh	12
package.is-installed()	12
File lib/output.sh	13
section()	13
Arguments	13
File lib/usage.sh	13
usage-widget()	13
Example	13
File lib/file-helpers.sh	14
.file.make_executable()	15
File lib/video.sh	15
lib/video.sh	15
File lib/path.sh	16
path.strip-slash()	17
path.dirs()	17
Arguments	17
path.dirs.size()	17
path.dirs.uniq()	17
path.dirs.delete()	17
Arguments	17
path.uniq()	17
PATH.uniqify()	17
path.append()	17
path.prepend()	18
path.mutate.uniq()	18
path.mutate.delete()	18
path.mutate.append()	18
path.mutate.prepend()	18
PATH_add()	18
File lib/osx.sh	18
osx.app.is-installed()	18
Example	18

Arguments .....	19
Exit codes .....	19
File <code>lib/db.sh</code> .....	19
<code>db.config.parse()</code> .....	19
Example .....	19
<code>db.psql.connect()</code> .....	20
Example .....	20
<code>db.psql.connect.just-data()</code> .....	20
Example .....	20
<code>db.psql.connect.table-settings-set()</code> .....	20
Example .....	20
<code>db.psql.db-settings()</code> .....	20
Example .....	20
<code>db.psql.connect.db-settings-pretty()</code> .....	20
Example .....	20
Arguments .....	21
<code>db.psql.connect.db-settings-toml()</code> .....	21
Example .....	21
Arguments .....	21
<code>db.actions.run-multiple()</code> .....	21
Example .....	21
<code>db.actions.pga()</code> .....	21
File <code>lib/shdoc.sh</code> .....	21
<code>lib/shdoc.sh</code> .....	21
File <code>lib/git.sh</code> .....	22
<code>git.cfgu()</code> .....	22
Example .....	22
<code>git.open()</code> .....	22
Example .....	22
Arguments .....	22
File <code>lib/package.sh</code> .....	22
<code>package.ensure.is-installed()</code> .....	23
<code>package.ensure.commmmand-available()</code> .....	23
Example .....	23
File <code>lib/time.sh</code> .....	23
<code>time.with-duration.start()</code> .....	23
Example .....	23
File <code>lib/shasum.sh</code> .....	23
<code>shasum.set-command()</code> .....	24
<code>shasum.set-algo()</code> .....	24
Example .....	24

shasum.sha()	24
shasum.sha-only()	24
shasum.sha-only-stdin()	24
shasum.to-hash()	24
Example	24
shasum.all-files()	24
Example	25
shasum.all-files-in-dir()	25
Example	25
File lib/runtime-config.sh	25
is.dry-run.on()	25
Example	25
is.dry-run.off()	25
Example	25
set.dry-run.on()	26
Example	26
set.dry-run.off()	26
Example	26
File lib/color.sh	26
color.current-background()	26
File lib/pg.sh	26
pg.is-running()	26
pg.running.server-binaries()	27
pg.running.data-dirs()	27
pg.server-in-path.version()	27
File lib/dir.sh	27
dir.short-home()	27
File lib/config.sh	27
config.get-format()	27
config.set-file()	27
config.get-file()	27
config.dig()	28
Arguments	28
config.dig.pretty()	28
File lib/flatten.sh	28
flatten-file()	28
Example	28
File lib/net.sh	28
net.is-host-port-protocol-open()	29
Arguments	29
File lib/is.sh	29

<code>__is.validation.error()</code> .....	29
Arguments .....	29
Exit codes .....	29
<code>is-validations()</code> .....	29
<code>__is.validation.ignore-error()</code> .....	29
<code>__is.validation.report-error()</code> .....	30
<code>whenever()</code> .....	30
Example .....	30
File <code>lib/util.sh</code> .....	30
File <code>lib/runtime.sh</code> .....	30
<code>run.inspect-vars()</code> .....	30
File <code>lib/pdf.sh</code> .....	30
Bashmatic Utilities for PDF file handling .....	30
File <code>bin/install-direnv</code> .....	31
<code>direnv.register()</code> .....	31
File <code>bin/regen-usage-docs</code> .....	31
File <code>bin/pdf-reduce</code> .....	31
<code>pdf.do.shrink()</code> .....	31
File <code>bin/scheck</code> .....	31
<code>manual-install()</code> .....	32
Copyright & License .....	32

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

---

## File `lib/file.sh`

- [file.temp\(\)](#)
- [file.normalize-files\(\)](#)
- [file.first-is-newer-than-second\(\)](#)
- [file.ask-if-exists\(\)](#)
- [file.install-with-backup\(\)](#)

### `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 copy 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 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"</format></format>`

### **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](http://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/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.

- [.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.uniqify\(\)](#)
- [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.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.

---

# 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 insensitively) 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.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.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/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/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/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/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.

---

## File `lib/runtime.sh`

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

## `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 © 2017-2021 Konstantin Gredeskoul, All rights reserved.
- Distributed under the MIT License.