Liquidprompt

Mark Vander Stel

Dec 21, 2020

CONTENTS:

1 Installation **3**
- 1.1 Packages 3
- 1.2 Download 5
- 1.3 Dependencies 5
- 1.4 Test Drive 6
- 1.5 Shell Installation 6

2 Config Options **7**
- 2.1 General 8
- 2.2 Features 8
- 2.3 Thresholds 12
- 2.4 Marks 13
- 2.5 Colors 14

3 Theming **19**
- 3.1 Default Theme 19
- 3.2 Included Themes 22
- 3.3 Custom Themes 29
- 3.4 Switching Themes 31

4 Functions **33**
- 4.1 Public Functions 33
- 4.2 Data Functions 35
- 4.3 Default Theme Functions 48
- 4.4 Utility Functions 51
- 4.5 Internal Functions 52

5 Release Notes **57**
- 5.1 Version 2.0 Release Notes 57
- 5.2 Version 1.12 Release Notes 59

6 Upgrading Liquidprompt **61**
- 6.1 Version 2.0 Upgrade Notes 61

7 Indices and tables **77**

Index **79**

Liquidprompt is an adaptive prompt for Bash & Zsh that gives you a nicely displayed prompt with useful information when you need it. It does this with a powerful theming engine and a large array of data sources.

To get started, view the *Installation* documentation, which includes instructions for trying Liquidprompt temporarily.

CHAPTER ONE

INSTALLATION

- *Download*
  - *Installation via Antigen*
- *Dependencies*
- *Test Drive*
- *Shell Installation*

1.1 Packages

- *Latest Versions*
- *Install commands*
  - *Archlinux*
  - *Debian*
  - *Homebrew*
  - *MX Linux*
  - *Nix*
  - *Ubuntu*

Liquidprompt is packaged for many operating systems, though the latest version in those repositories is not always up to date.

1.1.1 Latest Versions

Source: repology.org.

1.1.2 Install commands

Archlinux

```
pacman -S liquidprompt
```

Debian

```
apt-get install liquidprompt
```

Homebrew

```
brew install liquidprompt
```

MX Linux

```
apt-get install liquidprompt
```

Nix

```
nix-env -i liquidprompt
```

Ubuntu

```
apt-get install liquidprompt
```

1.2 Download

You can either download the latest release from Github, or using your OS package manager with our *Packages*.

To download to `~/liquidprompt`, run:

```
git clone https://github.com/nojhan/liquidprompt.git ~/liquidprompt
```

If you do not have `git`, you can download and extract the source in zip or gzip format directly from the release page.

1.2.1 Installation via Antigen

To install via Antigen, simply add the following line in your `.zshrc` after activating Antigen:

```
antigen bundle nojhan/liquidprompt
```

1.3 Dependencies

Liquidprompt uses commands that should be available on a large variety of Unix systems:

- `awk`
- `grep`
- `logname`
- `ps`
- `sed`
- `uname`
- `who`

Some features depend on specific commands. If you do not install them, the corresponding feature will not be available, but no error will be displayed. See the *Config Options* for more information about available features and what tools they require.

- Battery status requires `acpi` on GNU/Linux.
- Temperature status requires `acpi` or `sensors` on GNU/Linux.
- Terminal formatting requires `tput`.
- Detached session status looks for `screen` and/or `tmux`.
- VCS support features require `git`, `hg`, `svn`, `bzr` or `fossil` for their respective repositories.

1.4 Test Drive

To test the prompt immediately after download, run:

```
source ~/liquidprompt/liquidprompt
```

Adjust the path if you installed to a different location that the suggested `~/liquidprompt`.

1.5 Shell Installation

To use Liquidprompt every time you start a shell, add the following lines to your `.bashrc` (if you use Bash) or `.zshrc` (if you use zsh):

```
# Only load Liquidprompt in interactive shells, not from a script or from scp
[[ $- = *i* ]] && source ~/liquidprompt/liquidprompt
```

Adjust the path if you installed to a different location that the suggested `~/liquidprompt`.

Warning: Check in your `.bashrc` that the `PROMPT_COMMAND` variable is not set, or else the prompt will not be available. If you must set it or use a add-on that sets it, make sure to set `PROMPT_COMMAND` **before** you source Liquidprompt to avoid history and timing issues. Do not export `PROMPT_COMMAND`.

Next up are the *Config Options*.

CHAPTER

TWO

CONFIG OPTIONS

- *General*
- *Features*
- *Thresholds*
- *Marks*
- *Colors*

Almost every feature in Liquidprompt can be turned on or off using these config options. They can either be set before sourcing Liquidprompt (in `.bashrc` or `.zshrc`), or set in a Liquidprompt config file.

Note: Config variables set in a config file take precedence over variables set in the environment or on the command line. Setting a config option on the command line, then running *`lp_activate()`* will overwrite that option with the value from the config file, if it is set there.

The config file is searched for in the following locations:

- `~/.liquidpromptrc`
- `$XDG_CONFIG_HOME/liquidpromptrc` - (if `XDG_CONFIG_HOME` is not set, `~/.config` is used)
- `$XDG_CONFIG_DIRS/liquidpromptrc` - `XDG_CONFIG_DIRS` is a `:` delimited array, each value is searched. (if `XDG_CONFIG_DIRS` is not set, `/etc/xdg` is used)
- `/etc/liquidpromptrc`

The first file found is sourced.

Liquidprompt ships with an example config file, `liquidpromptrc-dist`. You can start from this file for your config:

```
cp ~/liquidprompt/liquidpromptrc-dist ~/.config/liquidpromptrc
```

Note: The example config file does not include every config option, and the comments describing the options are less verbose than the descriptions on this page.

Each config option is documented with its default value. Options of type `bool` accept values of `1` for true and `0` for false.

2.1 General

`LP_MARK_PREFIX: string = " "`

String added directly before *`LP_MARK_DEFAULT`*, after all other parts of the prompt. Can be used to tag the prompt in a way that is less intrusive than *`LP_PS1_PREFIX`*.

`LP_PATH_DEFAULT: string = '\w' (Bash) or '%~' (Zsh)`

Used to define the string used for the path. Could be used to make use of shell path shortening features, like `%2~` in Zsh to keep the last two directories of the path.

`LP_ENABLE_SHORTEN_PATH` must be disabled to have any effect.

`LP_PATH_KEEP: int = 2`

The number of directories (including '/') to display at the beginning of a shortened path.

Set to `1`, will display only root. Set to `0`, will keep nothing from the beginning of the path.

Set to `-1`, will display only the last directory of the path.

`LP_ENABLE_SHORTEN_PATH` must be enabled to have any effect.

See also: *`LP_PATH_LENGTH`*.

`LP_PATH_LENGTH: int = 35`

The maximum percentage of the terminal width used to display the path before removing the center portion of the path and replacing with *`LP_MARK_SHORTEN_PATH`*.

`LP_ENABLE_SHORTEN_PATH` must be enabled to have any effect.

See also: *`LP_PATH_KEEP`*.

`LP_PS1_POSTFIX: string = ""`

A string displayed at the very end of the prompt, after even the prompt mark. *`LP_MARK_PREFIX`* is an alternative that goes before the prompt mark.

`LP_PS1_PREFIX: string = ""`

A string displayed at the start of the prompt. Can also be set with *`prompt_tag()`*.

2.2 Features

`LP_DISABLED_VCS_PATH: string = ""`

Deprecated since version 2.0: Use *`LP_DISABLED_VCS_PATHS`* instead.

An colon (`:`) separated list of absolute directory paths where VCS features will be disabled. See *`LP_DISABLED_VCS_PATHS`* for more information.

`LP_DISABLED_VCS_PATHS: array<string> = ()`

An array of absolute directory paths where VCS features will be disabled. Generally this would be used for repositories that are large and slow, where generating VCS information for the prompt would impact prompt responsiveness.

Any subdirectory under the input directory is also disabled, so setting "/repos" would disable VCS display when the current directory is "/repos/a-repo". Setting `("/")` would disable VCS display completely.

An example value would be:

```
LP_DISABLED_VCS_PATHS=("/a/svn/repo" "/home/me/my/large/repo")
```

See also: *`LP_MARK_DISABLED`*.

LP_ENABLE_BATT: bool = 1

Display the status of the battery, if there is one, using color and marks. Add battery percentage colored with *LP_COLORMAP* if *LP_PERCENTS_ALWAYS* is enabled.

Will be disabled if `acpi` is not found on Linux, or `pmset` is not found on MacOS.

See also: *LP_BATTERY_THRESHOLD*, *LP_MARK_BATTERY*, *LP_MARK_ADAPTER*, *LP_COLOR_CHARGING_ABOVE*, *LP_COLOR_CHARGING_UNDER*, *LP_COLOR_DISCHARGING_ABOVE*, and *LP_COLOR_DISCHARGING_UNDER*.

LP_ENABLE_BZR: bool = 1

Display VCS information inside Bazaar repositories.

Will be disabled if `bzr` is not found.

See also: *LP_MARK_BZR*.

LP_ENABLE_COLOR: bool = 1

Use terminal formatting when displaying the prompt.

Note: Not all formatting is correctly disabled if this option is disabled.

Will be disabled if `tput` is not found.

New in version 2.0.

LP_ENABLE_DETACHED_SESSIONS: bool = 1

Display the number of detached multiplexer sessions.

Will be disabled if neither `screen` nor `tmux` are found.

Note: This can be slow on some machines, and prompt speed can be greatly improved by disabling it.

See also: *LP_COLOR_JOB_D*.

New in version 2.0.

LP_ENABLE_DIRSTACK: bool = 0

Display the size of the directory stack if it is greater than `1`.

See also: *LP_MARK_DIRSTACK* and *LP_COLOR_DIRSTACK*.

New in version 2.0.

LP_ENABLE_ERROR: bool = 1

Display the last command error code if it is not `0`.

See also: *LP_COLOR_ERR*.

New in version 2.0.

LP_ENABLE_FOSSIL: bool = 1

Display VCS information inside Fossil repositories.

Will be disabled if `fossil` is not found.

See also: *LP_MARK_FOSSIL*.

LP_ENABLE_FQDN: bool = 0

Use the fully qualified domain name (FQDN) instead of the short hostname when the hostname is displayed.

See also: *LP_HOSTNAME_ALWAYS*.

LP_ENABLE_GIT: bool = 1
Display VCS information inside Git repositories.

Will be disabled if `git` is not found.

See also: *LP_MARK_GIT*.

LP_ENABLE_HG: bool = 1
Display VCS information inside Mercurial repositories.

Will be disabled if `hg` is not found.

See also: *LP_MARK_HG*.

LP_ENABLE_JOBS: bool = 1
Display the number of running and sleeping shell jobs.

See also: *LP_COLOR_JOB_R* and *LP_COLOR_JOB_Z*.

LP_ENABLE_LOAD: bool = 1
Display the load average over the past 5 minutes when above the threshold. The value is expressed in centiload per CPU. For example, when using the default value of 60, the load average will be displayed starting at:

- 0.61 on a single-core machine
- 1.22 on a dual-core machine
- 2.44 on a quad-core machine, and so on.

We are aware this is not ideal, and welcome feedback and ideas on our issue for this topic: issue #530.

See also: *LP_LOAD_THRESHOLD*, *LP_MARK_LOAD*, and *LP_COLORMAP*.

LP_ENABLE_PERM: bool = 1
Display a colored *LP_MARK_PERM* in the prompt to show when the user does not have write permission to the current directory.

See also: *LP_COLOR_WRITE* and *LP_COLOR_NOWRITE*.

LP_ENABLE_PROXY: bool = 1
Display a *LP_MARK_PROXY* mark when an HTTP proxy is detected.

See also: *LP_COLOR_PROXY*.

LP_ENABLE_RUNTIME: bool = 1
Display runtime of the previous command if over *LP_RUNTIME_THRESHOLD*.

See also: *LP_COLOR_RUNTIME*.

LP_ENABLE_RUNTIME_BELL: bool = 0
Ring the terminal bell if the previous command ran longer than *LP_RUNTIME_BELL_THRESHOLD*.

New in version 1.12.

LP_ENABLE_SCLS: bool = 1
Display the currently activated Red Hat Software Collection.

See also: *LP_COLOR_VIRTUALENV*.

LP_ENABLE_SCREEN_TITLE: bool = 0
Set the terminal title while in a terminal multiplexer.

LP_ENABLE_TITLE must be enabled to have any effect.

LP_ENABLE_SHORTEN_PATH: bool = 1
Use the shorten path feature if the path is too long to fit in the prompt line.

See also: *LP_PATH_LENGTH*, *LP_PATH_KEEP*, and *LP_MARK_SHORTEN_PATH*.

LP_ENABLE_SSH_COLORS: bool = 0

Replace *LP_COLOR_SSH* with a color based on the hash of the hostname. This can give each host a "color feel" to help distinguish them.

See also: *LP_HOSTNAME_ALWAYS*.

LP_ENABLE_SUDO: bool = 0

Check if the user has valid `sudo` credentials, and display an indicating mark or color.

Will be disabled if `sudo` is not found.

> **Warning:** Each evocation of `sudo` by default writes to the syslog, and this will run `sudo` once each prompt. This is likely to make your sysadmin hate you.

See also: *LP_COLOR_MARK_SUDO*.

LP_ENABLE_SVN: bool = 1

Display VCS information inside Subversion repositories.

Will be disabled if `svn` is not found.

See also: *LP_MARK_SVN*.

LP_ENABLE_TEMP: bool = 1

Display the highest system temperature if above the threshold.

Will be disabled if neither `sensors` nor `acpi` are found.

See also: *LP_TEMP_THRESHOLD*, *LP_MARK_TEMP*, and *LP_COLORMAP*.

LP_ENABLE_TIME: bool = 0

Displays the time at which the prompt was shown.

See also: *LP_TIME_ANALOG* and *LP_COLOR_TIME*.

LP_ENABLE_TITLE: bool = 0

Set the terminal title to part or all of the prompt string, depending on the theme.

Must be enabled to be able to set the manual title with `lp_title()`.

> **Warning:** This may not work properly on exotic terminals. Please report any issues.

LP_ENABLE_VCS_ROOT: bool = 0

Enable VCS features when running as root. This is disabled by default for security.

LP_ENABLE_VIRTUALENV: bool = 1

Display the currently activated Python or Conda virtual environment.

See also: *LP_COLOR_VIRTUALENV*.

LP_HOSTNAME_ALWAYS: int = 0

Determine when the hostname should be displayed. Valid values are:

- `0` - show the hostname, except when locally connected
- `1` - always show the hostname
- `-1` - never show the hostname

See also: *LP_COLOR_HOST* and *LP_ENABLE_SSH_COLORS*.

LP_PERCENTS_ALWAYS: bool = 1
Display the percentages of load and batteries along with their corresponding marks. Disable to only print the colored marks.

LP_TIME_ANALOG: bool = 0
Shows the time using an analog clock instead of numeric values. The analog clock is "accurate" to the nearest half hour. You must have a unicode-capable terminal and a font with the "CLOCK" characters (U+1F550 - U+1F567).

Will only have an effect if *LP_ENABLE_TIME* is enabled.

LP_USER_ALWAYS: int = 1
Determine when the username should be displayed. Valid values are:

- `0` - show the username, except when the user is the login user
- `1` - always show the username
- `-1` - never show the username

See also: *LP_COLOR_USER_LOGGED*, *LP_COLOR_USER_ALT*, and *LP_COLOR_USER_ROOT*.

Changed in version 2.0: The `-1` option was added.

2.3 Thresholds

LP_BATTERY_THRESHOLD: int = 75
The percentage threshold that the battery level needs to fall below before it will be displayed in *LP_COLOR_CHARGING_UNDER* or *LP_COLOR_DISCHARGING_UNDER* color. Otherwise, it will be displayed in *LP_COLOR_CHARGING_ABOVE* or *LP_COLOR_DISCHARGING_ABOVE* color.

LP_ENABLE_BATT must be enabled to have any effect.

LP_LOAD_THRESHOLD: int = 60
Display the centiload average per CPU when above this threshold.

LP_ENABLE_LOAD must be enabled to have any effect.

LP_RUNTIME_THRESHOLD: int = 2
Time in seconds that a command must run longer than for its runtime to be displayed.

LP_ENABLE_RUNTIME must be enabled to have any effect.

LP_RUNTIME_BELL_THRESHOLD: int = 10
Time in seconds that a command must run longer than for the terminal bell to be rung.

LP_ENABLE_RUNTIME_BELL must be enabled to have any effect.

New in version 1.12.

LP_TEMP_THRESHOLD: int = 60
Display the highest system temperature when the temperature is above this threshold (in degrees Celsius).

LP_ENABLE_TEMP must be enabled to have any effect.

2.4 Marks

`LP_MARK_ADAPTER: string = ""`
Mark used for battery display when charging.

See also: *LP_ENABLE_BATT*.

`LP_MARK_BATTERY: string = ""`
Mark used for battery display when on battery power.

See also: *LP_ENABLE_BATT*.

`LP_MARK_BRACKET_CLOSE: string = "]"`
Mark used for closing core prompt brackets. Used by the default theme for enclosing user, host, and current working directory sections.

See also: *LP_MARK_BRACKET_OPEN*.

`LP_MARK_BRACKET_OPEN: string = "["`
Mark used for opening core prompt brackets. Used by the default theme for enclosing user, host, and current working directory sections.

See also: *LP_MARK_BRACKET_CLOSE*.

`LP_MARK_BZR: string = ""`
Mark used instead of *LP_MARK_DEFAULT* to indicate that the current directory is inside of a Bazaar repository.

See also: *LP_ENABLE_BZR*.

`LP_MARK_DEFAULT: string = "$" (Bash) or "%" (Zsh)`
Mark used to indicate that the prompt is ready for user input, unless some other context overrides it, like a VCS repository.

`LP_MARK_DIRSTACK: string = ""`
Mark used to indicate the size of the directory stack. Here are some alternative marks you might like: =

See also: *LP_ENABLE_DIRSTACK* and *LP_COLOR_DIRSTACK*.

New in version 2.0.

`LP_MARK_DISABLED: string = ""`
Mark used instead of *LP_MARK_DEFAULT* to indicate that the current directory is disabled for VCS display through *LP_DISABLED_VCS_PATHS*.

`LP_MARK_FOSSIL: string = ""`
Mark used instead of *LP_MARK_DEFAULT* to indicate that the current directory is inside of a Fossil repository.

See also: *LP_ENABLE_FOSSIL*.

`LP_MARK_GIT: string = "±"`
Mark used instead of *LP_MARK_DEFAULT* to indicate that the current directory is inside of a Git repository.

See also: *LP_ENABLE_GIT*.

`LP_MARK_HG: string = ""`
Mark used instead of *LP_MARK_DEFAULT* to indicate that the current directory is inside of a Mercurial repository.

See also: *LP_ENABLE_HG*.

`LP_MARK_LOAD: string = ""`
Mark used before displaying load average.

See also: *LP_ENABLE_LOAD*.

LP_MARK_PERM: string = ":"
Mark used by default separate hostname and current working directory, and is colored to indicate user permissions on the current directory.

Is still used (without colors) if *LP_ENABLE_PERM* is disabled.

New in version 1.12.

LP_MARK_PROXY: string = ""
Mark used to indicate a proxy is active.

See also: *LP_ENABLE_PROXY*.

LP_MARK_SHORTEN_PATH: string = " ... "
Mark used to indicate a portion of the path was hidden to save space.

See also: *LP_ENABLE_SHORTEN_PATH*.

LP_MARK_STASH: string = "+"
Mark used to indicate at least one stash or shelve exists in the current repository.

LP_MARK_SVN: string = "‡"
Mark used instead of *LP_MARK_DEFAULT* to indicate that the current directory is inside of a Subversion repository.

See also: *LP_ENABLE_SVN*.

LP_MARK_TEMP: string = ""
Mark used before displaying temperature.

See also: *LP_ENABLE_TEMP*.

LP_MARK_UNTRACKED: string = "*"
Mark used to indicate untracked or extra files exist in the current repository.

LP_MARK_VCSH: string = "|"
Mark used instead of *LP_MARK_DEFAULT* to indicate that the current directory is inside of a VCSH repository.

Since VCSH repositories are Git repositories under the hood, *LP_MARK_GIT* is surrounded in this mark.

2.5 Colors

These color strings will be used without modification, so they need to be valid terminal escape sequences, either generated with *lp_terminal_format()* or using the `$COLOR` variables.

Valid preset color variables are:

- `BOLD` - bold formatting only.
- `BLACK`
- `BOLD_GRAY` - actually bold black
- `RED`
- `BOLD_RED`
- `GREEN`
- `BOLD_GREEN`
- `YELLOW`
- `BOLD_YELLOW`

- `BLUE`
- `BOLD_BLUE`
- `PURPLE` or `MAGENTA`
- `BOLD_PURPLE`, `BOLD_MAGENTA` or `PINK`
- `CYAN`
- `BOLD_CYAN`
- `WHITE`
- `BOLD_WHITE`
- `WARN_RED` - black foreground, red background
- `CRIT_RED` - white foreground, red background
- `DANGER_RED` - yellow foreground, red background

LP_COLORMAP: array<string>
An array of colors that is used by the battery, load, and temperature features to indicate the severity level of their status. A normal or low status will use the first index, while the last index is the most severe.

The default array is:

```
(
    ""
    $GREEN
    $BOLD_GREEN
    $YELLOW
    $BOLD_YELLOW
    $RED
    $BOLD_RED
    $WARN_RED
    $CRIT_RED
    $DANGER_RED
)
```

See also: *LP_ENABLE_BATT*, *LP_ENABLE_LOAD*, and *LP_ENABLE_TEMP*.

LP_COLOR_CHANGES: string = $RED
Color used to indicate that the current repository is not clean, or in other words, has changes that have not been committed.

LP_COLOR_CHARGING_ABOVE: string = $GREEN
Color used to indicate that the battery is charging and above the *LP_BATTERY_THRESHOLD*.

See also: *LP_ENABLE_BATT*.

LP_COLOR_CHARGING_UNDER: string = $YELLOW
Color used to indicate that the battery is charging and under the *LP_BATTERY_THRESHOLD*.

See also: *LP_ENABLE_BATT*.

LP_COLOR_COMMITS_BEHIND: string = $BOLD_RED
Color used to indicate that the current repository has a remote tracking branch that has commits that the local branch does not.

LP_COLOR_COMMITS: string = $YELLOW
Color used to indicate that the current repository has commits on the local branch that the remote tracking branch does not.

Also used to color *LP_MARK_STASH*.

`LP_COLOR_DIFF: string = $PURPLE`
Color used to indicate that the current repository has lines that have been changed since the last commit.

`LP_COLOR_DIRSTACK: string = $BOLD_YELLOW`
Color used to indicate the size of the directory stack.

See also: *LP_ENABLE_DIRSTACK* and *LP_MARK_DIRSTACK*.

New in version 2.0.

`LP_COLOR_DISCHARGING_ABOVE: string = $YELLOW`
Color used to indicate that the battery is discharging and above the *LP_BATTERY_THRESHOLD*.

See also: *LP_ENABLE_BATT*.

`LP_COLOR_DISCHARGING_UNDER: string = $RED`
Color used to indicate that the battery is discharging and above the *LP_BATTERY_THRESHOLD*.

See also: *LP_ENABLE_BATT*.

`LP_COLOR_ERR: string = $PURPLE`
Color used to indicate the last command exited with a non-zero return code.

See also: *LP_ENABLE_ERROR*.

`LP_COLOR_HOST: string = ""`
Color used for the hostname when connected locally.

See also: *LP_HOSTNAME_ALWAYS*.

`LP_COLOR_IN_MULTIPLEXER: string = $BOLD_BLUE`
Color used for *LP_MARK_BRACKET_OPEN* and *LP_MARK_BRACKET_CLOSE* if the terminal is in a multiplexer.

`LP_COLOR_JOB_D: string = $YELLOW`
Color used for detached multiplexer sessions.

See also: *LP_ENABLE_DETACHED_SESSIONS*.

`LP_COLOR_JOB_R: string = $BOLD_YELLOW`
Color used for running shell jobs.

See also: *LP_ENABLE_JOBS*.

`LP_COLOR_JOB_Z: string = $BOLD_YELLOW`
Color used for sleeping shell jobs.

See also: *LP_ENABLE_JOBS*.

`LP_COLOR_MARK: string = $BOLD`
Color used for *LP_MARK_DEFAULT*.

`LP_COLOR_MARK_ROOT: string = $BOLD_RED`
Color used for *LP_MARK_DEFAULT* when the current user is root, shown instead of *LP_COLOR_MARK*.

`LP_COLOR_MARK_SUDO: string = $LP_COLOR_MARK_ROOT`
Color used for *LP_MARK_DEFAULT* when sudo is active, shown instead of *LP_COLOR_MARK*.

See also: *LP_ENABLE_SUDO*.

`LP_COLOR_NOWRITE: string = $RED`
Color used for *LP_MARK_PERM* when the user does not have write permissions to the current working directory.

See also: *LP_ENABLE_PERM* and *LP_COLOR_WRITE*.

LP_COLOR_PATH: string = $BOLD
Color used for the current working directory.

LP_COLOR_PATH_ROOT: string = $BOLD_YELLOW
Color used in place of *LP_COLOR_PATH* when the current user is root.

LP_COLOR_PROXY: string = $BOLD_BLUE
Color used for *LP_MARK_PROXY*.

See also: *LP_ENABLE_PROXY*.

LP_COLOR_RUNTIME: string = $YELLOW
Color used for displaying the last command runtime.

See also: *LP_ENABLE_RUNTIME*.

LP_COLOR_SSH: string = $BLUE
Color used for displaying the hostname when connected with SSH.

Has no effect if *LP_ENABLE_SSH_COLORS* is enabled.

See also: *LP_HOSTNAME_ALWAYS*.

LP_COLOR_SU: string = $BOLD_YELLOW
Color used for displaying the hostname when running under `su` or `sudo`.

See also: *LP_HOSTNAME_ALWAYS*.

LP_COLOR_TELNET: string = $WARN_RED
Color used for displaying the hostname when connected with Telnet.

See also: *LP_HOSTNAME_ALWAYS*.

LP_COLOR_TIME: string = $BLUE
Color used for displaying the current time.

See also: *LP_ENABLE_TIME*.

LP_COLOR_UP: string = $GREEN
Color used to indicate that the current repository is up-to-date and no commits differ from the remote tracking branch.

LP_COLOR_USER_ALT: string = $BOLD
Color used for displaying the username when running as a different user than the login user.

LP_COLOR_USER_LOGGED: string = ""
Color used for displaying the username when running as the login user.

See also: *LP_USER_ALWAYS*.

LP_COLOR_USER_ROOT: string = $BOLD_YELLOW
Color used for displaying the username when running as root.

LP_COLOR_VIRTUALENV: string = $CYAN
Color used for displaying a Python virtual env or Red Hat Software Collection.

See also: *LP_ENABLE_VIRTUALENV* and *LP_ENABLE_SCLS*.

LP_COLOR_WRITE: string = $GREEN
Color used for *LP_MARK_PERM* when the user has write permissions to the current working directory.

See also: *LP_ENABLE_PERM* and *LP_COLOR_NOWRITE*.

LP_COLOR_X11_OFF: string = $YELLOW
Color used for indicating that a display is not connected.

`LP_COLOR_X11_ON: string = $GREEN`

Color used for indicating that a display is connected.

CHAPTER

THREE

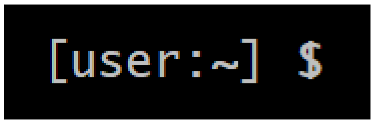
THEMING

3.1 Default Theme

- *Preview*
- *Configuation*
- *Templates*
  - *Template Sections*

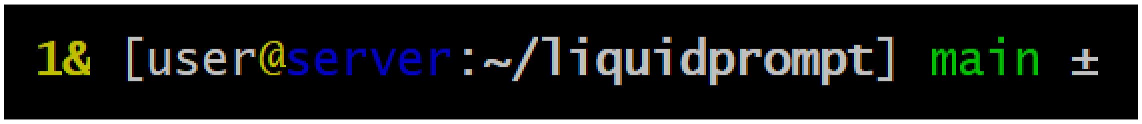
3.1.1 Preview

If there is nothing special about the current context, the appearance of Liquidprompt is similar to that of a default prompt:



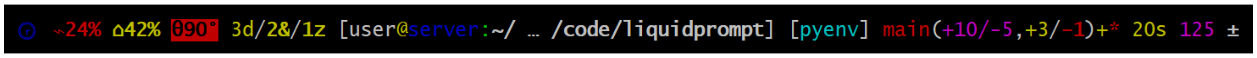
```
[user:~] $
```

If you are running a background command and are also in the “main” branch of a Git repository on a server:



```
1& [user@server:~/liquidprompt] main ±
```

When Liquidprompt is displaying nearly everything (a rare event!), it may look like this:



```
⊙ ⌁24% ⌂42% θ90° 3d/2&/1z [user@server:~/ … /code/liquidprompt] [pyenv] main(+10/-5,+3/-1)+* 20s 125 ±
```

See *Templates* for what each section will look like.

3.1.2 Configuation

As the default theme, all of the normal *Config Options* are respected.

`LP_PS1_FILE: string = ""`
A template file that is sourced for each prompt. Must set *`LP_PS1`*. See *Templates* for details.

`LP_PS1: string = ""`
If set, the default theme sets `PS1` to this value. Not very useful to set it in the config, instead set it in the *`LP_PS1_FILE`*.

3.1.3 Templates

The default theme supports templated sections. Each piece of the theme is saved to a variable, and can be arranged in any order in a template. If you want to change the theme enough to move things around, but not enough to make your own theme, templates will let you change the order of the default theme's pieces.

As the default theme of Liquidprompt was the only theme until version 2.0, templates were sometimes referred to as "themes" in version 1.X.

For a template file to be loaded, its filepath must be set in *`LP_PS1_FILE`*.

A template file does nothing more than set *`LP_PS1`* to a value. The following sections are available to be used.

An example template file is available: liquid.ps1.

Template Sections

All of the available template sections are listed below. Their order is the default order if the user does not configure a different template.

Note: Omitting a template section from your template will **not** disable that feature. While it will not be displayed in the prompt, Liquidprompt does not know that, and will still generate that template section. If you want to speed up your prompt by disabling a section, you must disable it with its respective `LP_ENABLE_*` option.

`LP_PS1_PREFIX`:

Not actually a part of the default theme, it is used in the default template as the starting section. See *`LP_PS1_PREFIX`* and *`prompt_tag()`* for details.

`LP_TIME`
The current time, displayed as either numeric values or as an analog clock, depending on the value of *`LP_TIME_ANALOG`*. See *`LP_ENABLE_TIME`*.

`LP_BATT`
The current battery status:

- a green (*`LP_MARK_BATTERY`*) if charging, above the given threshold, but not charged
- a yellow if charging and under the given threshold
- a yellow (*`LP_MARK_ADAPTER`*) if discharging but above the given threshold
- a red if discharging and under the given threshold

And if *`LP_PERCENTS_ALWAYS`* is enabled, also the current battery percent. See *`LP_ENABLE_BATT`*.

LP_LOAD
The average of the processors load, displayed with an intensity color map as load increases. See *LP_ENABLE_LOAD*.

LP_TEMP
The highest temperature of the available system sensors, displayed with an intensity color map as temperature increases. See *LP_ENABLE_TEMP*.

LP_JOBS
The number of detached sessions. See *LP_ENABLE_DETACHED_SESSIONS*.

Also the number of running and sleeping shell jobs. See *LP_ENABLE_JOBS*.

LP_BRACKET_OPEN
An opening bracket, designed to go around the core of the prompt (generally user, host, current working directory). See *LP_MARK_BRACKET_OPEN*.

If running in a terminal multiplexer, will be colored. See *LP_COLOR_IN_MULTIPLEXER*.

LP_USER
The current user, in bold yellow if it is root and in light white if it is not the same as the login user. See *LP_USER_ALWAYS*.

LP_HOST
A green `@` if the connection has X11 support; a yellow one if not.

The current host – in bold red if you are connected via a `telnet` connection and blue (or other unique colors) if connected via SSH. See *LP_HOSTNAME_ALWAYS*.

LP_PERM
A green colon (*LP_MARK_PERM*) if the user has write permissions in the current directory and a red one if not. See *LP_ENABLE_PERM*.

LP_PWD
The current working directory in bold, shortened if it takes too much space. See *LP_ENABLE_SHORTEN_PATH*.

LP_DIRSTACK
The size of the directory stack, prefixed with *LP_MARK_DIRSTACK*, all colored with *LP_COLOR_DIRSTACK*. Can be enabled by *LP_ENABLE_DIRSTACK*.

LP_BRACKET_CLOSE
A closing bracket, designed to go around the core of the prompt (generally user, host, current working directory). See *LP_MARK_BRACKET_CLOSE*.

If running in a terminal multiplexer, will be colored. See *LP_COLOR_IN_MULTIPLEXER*.

LP_SCLS
The current Red Hat Software Collections environment. See *LP_ENABLE_SCLS*.

LP_VENV
The current Python (or Conda) virtual environment. See *LP_ENABLE_VIRTUALENV*.

LP_PROXY
A (*LP_MARK_PROXY*) if an HTTP proxy is in use. See *LP_ENABLE_PROXY*.

LP_VCS

- The name of the current branch if you are in a version control repository (Git, Mercurial, Subversion, Bazaar, or Fossil):
  - in green if everything is up-to-date
  - in red if there are changes

  - in yellow if there are pending commits to push
- The number of added/deleted lines if changes have been made and the number of pending commits
- The number of commits ahead/behind the remote tracking branch
- A yellow + (*LP_MARK_STASH*) if there are stashed modifications
- a red * (*LP_MARK_UNTRACKED*) if there are untracked files in the repository

LP_RUNTIME
The runtime of the last command, if it has exceeded a certain threshold. See *LP_ENABLE_RUNTIME*.

LP_ERR
The error code of the last command, if it is non-zero. See *LP_ENABLE_ERROR*.

LP_MARK_PREFIX

Not actually a part of the default theme, it is used in the default template as the last thing before the prompt mark. See *LP_MARK_PREFIX* for details.

LP_COLOR_MARK

Bold normally, red if you have `sudo` rights or for the root user.

Separate from *LP_MARK* for historical reasons.

LP_MARK
A smart mark at the end of the prompt:

- $ or % (*LP_MARK_DEFAULT*) for a simple user
- # for the root user
- (*LP_MARK_FOSSIL*) for Fossil
- ± (*LP_MARK_GIT*) for Git
- (*LP_MARK_HG*) for Mercurial
- ‡ (*LP_MARK_SVN*) for Subversion
- ‡± for Git-Subversion
- |±| (*LP_MARK_VCSH*) for VCSH

LP_PS1_POSTFIX

Not actually a part of the default theme, it is used in the default template as the final section. See *LP_PS1_POSTFIX* for details.

3.2 Included Themes

Liquidprompt ships with some included themes that will have features added to them as they are added to Liquidprompt.

3.2.1 Powerline Theme

The included `themes/powerline/powerline.theme` file includes two themes:

- *Powerline*
  - *Preview*
  - *Setup*
  - *Configuation*
    - *Liquidprompt Configuration*
    - *Theme Configuration*
      - *Markers*
      - *Colors*
- *Powerline Full*
  - *Preview*
  - *Setup*
  - *Configuation*
    - *Liquidprompt Configuration*
    - *Theme Configuration*
      - *Markers*
      - *Colors*

Powerline

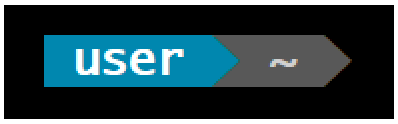
The `powerline` theme is a clone of the Powerline prompt. It copies the default segments of the Powerline prompt for Shell.

This prompt is a proof of (a specific) concept: that Liquidprompt can do what Powerline does, but faster. That said, this is a fully usable theme.

New in version 2.0.

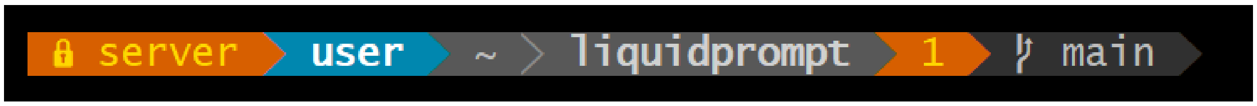
Preview

If there is nothing special about the current context, the appearance of Powerline might be as simple as this:



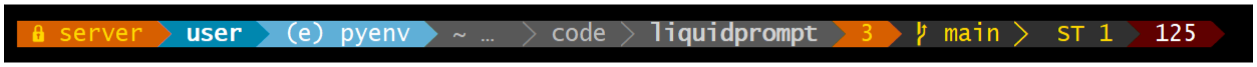
```
user ~
```

If you are running a background command and are also in the "main" branch of a Git repository on a server:



```
server user ~ liquidprompt 1 main
```

When Liquidprompt is displaying nearly everything, it may look like this:



```
🔒 server  user  (e) pyenv  ~ … code  liquidprompt  3  main  ST 1  125
```

Note: The above "everything" image looks like it is missing some parts because this theme does not implement all data sources of Liquidprompt. This is by design to clone basic Powerline. For a Powerline theme that does show all data sources, see *Powerline Full* below.

Setup

By default, the dividers and markers used are the Powerline private characters. You will either need a compatible font, or to configure the dividers and markers to use other characters.

See the Powerline Fonts installation docs for help.

Configuation

Liquidprompt Configuration

The following Liquidprompt config options are respected:

- *LP_DISABLED_VCS_PATHS*
- *LP_ENABLE_BZR*
- *LP_ENABLE_COLOR*
- *LP_ENABLE_ERROR*
- *LP_ENABLE_FOSSIL*
- *LP_ENABLE_FQDN*
- *LP_ENABLE_GIT*
- *LP_ENABLE_HG*
- *LP_ENABLE_JOBS*
- *LP_ENABLE_RUNTIME_BELL*
- *LP_ENABLE_SCREEN_TITLE*
- *LP_ENABLE_SHORTEN_PATH*
- *LP_ENABLE_SVN*
- *LP_ENABLE_TITLE*
- *LP_ENABLE_VCS_ROOT*
- *LP_ENABLE_VIRTUALENV*
- *LP_HOSTNAME_ALWAYS*
- *LP_PATH_DEFAULT*
- *LP_PATH_KEEP*
- *LP_PATH_LENGTH*

- *LP_RUNTIME_BELL_THRESHOLD*
- *LP_USER_ALWAYS*

Theme Configuration

Powerline adds these config options:

Markers

POWERLINE_HARD_DIVIDER: string = "" # U+E0B0
The divider character between sections, defaults to the private character used in Powerline fonts that looks like a solid right arrow.

POWERLINE_PYTHON_ENV_MARKER: string = "(e) "
The marker string used to indicate the following string is a Python environment.

POWERLINE_ROOT_MARKER: string = "#"
The marker character used to indicate a root session.

POWERLINE_SECURE_MARKER: string = "" # U+E0A2
The marker character used to indicate a SSH session, defaults to the private character used in Powerline fonts that looks like a lock.

POWERLINE_SOFT_DIVIDER: string = "" # U+E0B1
The divider character between similar sections, defaults to the private character used in Powerline fonts that looks like a thin right arrow.

POWERLINE_SPACER: string = " " # U+00A0: non-breaking space
The marker character used to pad sections, defaults to the non-breaking space character.

To add more padding, add more spaces to this string.

A non-breaking space is needed in some fonts to prevent multiple spaces from collapsing to one space, loosing the padding.

POWERLINE_STASH_MARKER: string = "ST"
The marker string used to indicate stashes exist in the VCS repository.

POWERLINE_VCS_MARKER: string = "" # U+E0A0
The marker character used to indicate a VCS repository, defaults to the private character used in Powerline fonts that looks like a branching commit history.

Colors

These color config options take an array of integers, which are arguments to *lp_terminal_format()*.

Note: Arrays are set without commas (`,`). The default values are displayed with commas for clarity.

POWERLINE_ERROR_COLOR: array<int> = (231, 52, 0, 0, 7, 1)
Color for the error code section.

POWERLINE_HOST_COLOR: array<int> = (220, 166, 0, 0, 3, 2)
Color for the hostname section.

`POWERLINE_JOBS_COLOR: array<int> = (220, 166, 0, 0, 3, 2)`
Color for the shell jobs section.

`POWERLINE_PATH_COLOR: array<int> = (250, 240, 0, 0, 7, 0)`
Color for the current working directory section.

`POWERLINE_PATH_LAST_COLOR: array<int> = (252, 240, 1, 0, 7, 0)`
Color for the current working directory last subsection.

`POWERLINE_PATH_SEPARATOR_COLOR: array<int> = (245, 240, 0, 0, 7, 0)`
Color for the current working directory subsection separator.

`POWERLINE_PYTHON_ENV_COLOR: array<int> = (231, 74, 0, 0, 7, 4)`
Color for the Python environment section.

`POWERLINE_USER_COLOR: array<int> = (231, 31, 1, 0, 7, 6)`
Color for the username section.

`POWERLINE_VCS_CLEAN_COLOR: array<int> = (250, 236, 0, 0, 7, 0)`
Color for the VCS section if the repository is clean.

`POWERLINE_VCS_DIRTY_COLOR: array<int> = (220, 236, 0, 0, 3, 0)`
Color for the VCS section if the repository is not clean.

`POWERLINE_VCS_STASH_COLOR: array<int> = (220, 236, 0, 0, 3, 0)`
Color for the VCS stash subsection.

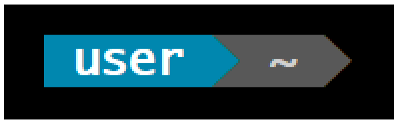
Powerline Full

An extension of the `powerline` theme, `powerline_full` includes all data sources that Liquidprompt provides. The ordering is the same as the default theme.

New in version 2.0.

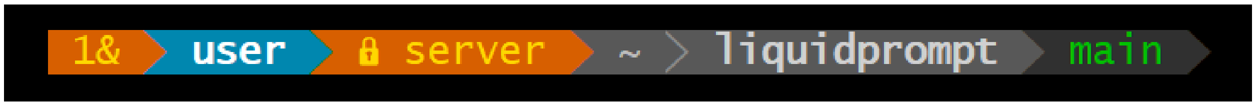
Preview

If there is nothing special about the current context, the appearance of Powerline might be as simple as this:



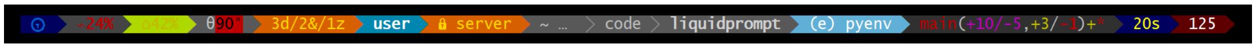
```
user ~
```

If you are running a background command and are also in the "main" branch of a Git repository on a server:



```
1& user 🔒 server ~ liquidprompt main
```

When Liquidprompt is displaying nearly everything, it may look like this:



```
⊙ ⌁24% ⌂42% θ90° 3d/2&/1z user 🔒 server ~ … code liquidprompt (e) pyenv main(+10/-5,+3/-1)+* 20s 125
```

Setup

Like the `powerline` theme, you will need a compatible font. See the Powerline Fonts installation docs for help.

Configuation

Liquidprompt Configuration

All Liquidprompt config options are respected, **except for**:

- *LP_COLOR_DIRSTACK*
- *LP_COLOR_ERR*
- *LP_COLOR_HOST*
- *LP_COLOR_IN_MULTIPLEXER*
- *LP_COLOR_JOB_D*
- *LP_COLOR_JOB_R*
- *LP_COLOR_JOB_Z*
- *LP_COLOR_MARK*
- *LP_COLOR_MARK_ROOT*
- *LP_COLOR_MARK_SUDO*
- *LP_COLOR_NOWRITE*
- *LP_COLOR_PATH*
- *LP_COLOR_PATH_ROOT*
- *LP_COLOR_PROXY*
- *LP_COLOR_RUNTIME*
- *LP_COLOR_SSH*
- *LP_COLOR_SU*
- *LP_COLOR_TELNET*
- *LP_COLOR_TIME*
- *LP_COLOR_USER_ALT*
- *LP_COLOR_USER_LOGGED*
- *LP_COLOR_USER_ROOT*
- *LP_COLOR_VIRTUALENV*
- *LP_COLOR_WRITE*
- *LP_COLOR_X11_OFF*
- *LP_COLOR_X11_ON*
- *LP_ENABLE_PERM*
- *LP_ENABLE_SSH_COLORS*
- *LP_ENABLE_SUDO*

- *LP_MARK_BRACKET_OPEN*
- *LP_MARK_BRACKET_CLOSE*
- *LP_MARK_BZR*
- *LP_MARK_DEFAULT*
- *LP_MARK_DISABLED*
- *LP_MARK_FOSSIL*
- *LP_MARK_GIT*
- *LP_MARK_HG*
- *LP_MARK_PERM*
- *LP_MARK_PREFIX*
- *LP_MARK_PROXY*
- *LP_MARK_SVN*
- *LP_MARK_VCSH*

Theme Configuration

Powerline Full uses all the config options of the above Powerline theme, **except for**:

- *POWERLINE_STASH_MARKER*
- *POWERLINE_VCS_DIRTY_COLOR*
- *POWERLINE_VCS_MARKER*
- *POWERLINE_VCS_STASH_COLOR*

Powerline Full adds these config options:

Markers

`POWERLINE_CHROOT_MARKER: string = "chroot:  "`
    The marker string used to indicate the following string is a chroot.

`POWERLINE_PROXY_MARKER: string = "proxy:  "`
    The marker string used to indicate the following string is a HTTP proxy.

`POWERLINE_SOFTWARE_COLLECTION_MARKER: string = "(sc) "`
    The marker string used to indicate the following string is a Red Hat Software Collection.

Colors

`POWERLINE_BATTERY_COLOR: array<int> = (-1, 238, 0, 0, -1, 0)`
Color for the battery section.

`POWERLINE_CHROOT_COLOR: array<int> = (219, 30, 0, 0, 7, 4)`
Color for the chroot section.

`POWERLINE_DIRSTACK_COLOR: array<int> = $POWERLINE_NEUTRAL_COLOR`
Color for the directory stack section.

`POWERLINE_LOAD_COLOR: array<int> = (-1, 148, 0, 0, -1, 3)`
Color for the CPU load section.

`POWERLINE_NEUTRAL_COLOR: array<int> = (252, 234, 0, 0, 7, 0)`
Color for all neutral sections, *`LP_PS1_PREFIX`* and *`LP_PS1_POSTFIX`*.

`POWERLINE_PROXY_COLOR: array<int> = (21, 219, 1, 0, 4, 7)`
Color for the HTTP proxy section.

`POWERLINE_RUNTIME_COLOR: array<int> = (226, 17, 0, 0, 3, 4)`
Color for the command runtime section.

`POWERLINE_SOFTWARE_COLLECTIONS_COLOR: array<int> = (231, 62, 0, 0, 7, 5)`
Color for the Red Hat Software Collections section.

`POWERLINE_TEMPERATURE_COLOR: array<int> = (-1, 240, 0, 0, -1, 0)`
Color for the temperature section.

`POWERLINE_TIME_COLOR: array<int> = (33, 17, 0, 0, 5, 4)`
Color for the current time section.

3.3 Custom Themes

- *Defining a Theme*
  - *Prompt Function*
  - *Directory Function*
  - *Activate Function*
  - *Other Functions*
- *Getting Data*
- *Examples*
- *Sharing Your Theme*

3.3.1 Defining a Theme

A theme should be contained in one file with a `.theme` file suffix. There should be no "top level" code in the file, or in other words, all code should be contained in functions. Sourcing the file should run no code, as a user sourcing the theme file might not want to activate it yet.

Prompt Function

Every theme must have a prompt function that is called for every prompt to generate the prompt. It *must* be set to `_lp_<theme_id>_theme_prompt()`.

This function could do anything, but generally it should generate a prompt and store it in `PS1`.

Directory Function

Optionally, a theme can have a directory function. It must be set to `_lp_<theme_id>_theme_directory()`.

This function is called every time the user changes directories. This allows the theme to only run generating code that depends on the current directory when it is needed.

Activate Function

Optionally, a theme can have an activate function. It must be set to `_lp_<theme_id>_theme_activate()`.

This function is called when the theme is first activated, and every time the user runs *`lp_activate()`*. Prompt pieces that never change (such as hostname and username) should be generated here. This is also where the theme's default values should be set. This function will always be called after the user config is already loaded.

Other Functions

If a theme is moderately complicated, it will need other functions defined to help generate a prompt. These should be named following the *Functions* guidelines concerning underscore prefixes.

The prefix of a function should always be either `_<theme_id>_` or `_lp_<theme_id>_` to prevent overwriting functions already defined by the user.

3.3.2 Getting Data

A theme must call *Data Functions* to be able to display useful information to the user. A theme might also need to use *Utility Functions* to process that data.

3.3.3 Examples

The *Powerline Theme* is a good example of creating a detailed theme.

3.3.4 Sharing Your Theme

First see the Theme sharing wiki page for things you should do to make your theme shareable.

The Themes wiki page is where you can share your theme with other users.

- *Switching Themes*

Liquidprompt has a strong data and theming engine, allowing it to be extremely flexible and customizable.

The *Default Theme* has a templating engine (previously called "themes" in Liquidprompt version 1), that allows for custom prompt ordering in the default theme.

Liquidprompt ships with some *Included Themes* other than the default as well.

See the Liquidprompt Theme List on the wiki for user created themes.

If you want to create your own theme, see *Custom Themes*.

3.4 Switching Themes

Liquidprompt can switch between themes on the fly. The shell does not need to be reloaded, and no files need to be sourced after the initial source.

To load (but not activate) a theme, simply source the theme file. For example, to load the included Powerline theme, source the theme file:

```
$ source themes/powerline/powerline.theme
```

Now both the default theme and Powerline are loaded. To show what themes are loaded and available, run `lp_theme()`:

```
$ lp_theme --list
default
powerline_full
powerline
```

To switch to a different theme, call `lp_theme()` with the name of the theme as the argument:

```
$ lp_theme powerline
```

The prompt will immediately take on the new theme.

To switch back to the default theme, call `lp_theme()` again with `default` as the argument instead.

If you add the theme source commands to your shell startup file, you will have your favorite themes ready to be switched to at any time.

CHAPTER

FOUR

FUNCTIONS

Functions starting with `lp` or any other alphanumeric character are **public** functions designed to be used by users on the command line or in their config.

Functions starting with `_lp` are **theme** level functions, designed to be used by themes. These include data, theme, and utility functions.

Functions starting with `__lp` are **internal** functions, designed to be used only by Liquidprompt internals. These functions should not be used by users or themes, as they are not guaranteed to not change between versions.

4.1 Public Functions

These functions are designed to be used by users on the command line or in their config.

`lp_activate``()`

Reload the user config.

This function is called when sourcing `liquidprompt`, unless the flag `--no-activate` is passed.

The config is sourced, and the environment scanned again for programs needed for specific features.

Lastly, *`prompt_on()`* is called to enabled the prompt.

New in version 2.0.

`lp_title``(`[*title_string*]`)`

Set *title_string* as the terminal title. This overrides any title set by the current theme.

Note: The input string is not escaped in any way; if it contains characters that the shell will interpret, the user must escape them if that behavior is not desired.

To unset the manual title, call *`lp_title()`* with no arguments.

To set a blank title, call *`lp_title()`* with an empty string argument (`''`).

This function will do nothing and return `2` if *`LP_ENABLE_TITLE`* is `0`.

New in version 2.0.

`lp_theme``(`*theme_id* | *--list*`)`

Load and activate the theme named *theme_id*. The theme functions must be loaded into memory before *`lp_theme()`* can be called, normally by sourcing the theme file.

The optional flag `--list` will instead list all currently loaded themes.

This function supports shell autocompletion.

New in version 2.0.

`lp_terminal_format` (*foreground_color*[, *background_color*][, *bold*][, *underline*][, *fallback_foreground_color*][, *fallback_background_color*]) → var:lp_terminal_format

Generate a shell escaped terminal formatting string for use in `PS1`.

The start of the formatting string always resets back to terminal defaults.

foreground_color and *background_color* accept an ANSI escape color code integer to set the color of the foreground and background, respectively. The behavior depends on the integer:

- `>= 0 && < max_color` - The color is used directly.
- `>= max_color` - If the terminal reports that the number of colors it supports is less than the input color code, the *fallback_foreground_color* or *fallback_background_color* is used instead.
- `-1` - No color is set. This does not mean that the previous color will continue over, as all formatting is reset to default at the start of the sequence. This means the default coloring is effectively set.
- `-2` - The previous color of the field is set. If no color was previously set, no color will be set. Note that the output is a static formatting string; the string will not keep the same color as the terminal previously had, but the color that was last selected when *`lp_terminal_format()`* was last run.
- `-3` - Same as `-2`, except the opposite field color is copied. In other words, if *foreground_color* is set to `-3`, it will copy the color of *background_color* the last time *`lp_terminal_format()`* was run.

bold and *underline* enable their respective formats when set to `1`. If omitted or set to `0`, they are not enabled. To use fallback colors, they will need to be set to be able to set the other options.

fallback_foreground_color and *fallback_background_color* are used when the normal colors are higher than the terminal supported colors. The special negative inputs do not work for these options, and they are not checked for compatibility before being used, so it is recommended that they are in the range `0-7`. When setting *foreground_color* or *background_color* to negative inputs, these options are never checked. New in version 2.0:

`prompt_on`()

Enable the prompt generation and setting.

This function is called when sourcing `liquidprompt`, unless the flag `--no-activate` is passed.

`prompt_off`()

Disable the prompt generation and setting, and restore the old `PS1`.

If the shell is Bash, also restore the old `PROMPT_COMMAND`.

If the shell is Zsh, also restore the old prompt theme.

`prompt_OFF`()

Same as *`prompt_off()`*, except instead of restoring the previous `PS1`, it is set to "$ " on Bash, "% " on Zsh.

`prompt_tag`([*prefix_string*])

Sets a prefix that will be displayed before every prompt. Postpends a space to the input string.

Internally, this function sets *`LP_PS1_PREFIX`* to *prefix_string*. If a trailing space is not wanted, set *`LP_PS1_PREFIX`* manually.

To unset the prefix, call *`prompt_tag()`* with no arguments.

4.2 Data Functions

- *Battery*
- *Development Environment*
- *Environment*
- *Jobs*
- *Load*
- *OS*
- *Runtime*
- *Temperature*
- *Time*

4.2.1 Version Control Data Functions

- *Generic*
- *Bazaar*
- *Fossil*
- *Git*
- *Mercurial*
- *Subversion*

These functions are designed to be used by themes.

Generic

The generic interface functions are designed to provide a level of abstraction over the type of VCS that a user might be using. By using the generic interface, a theme can provide a common look for all VCS types. See the default theme function `_lp_vcs_details_color()` for an example of this.

`_lp_find_vcs``()` → var:lp_vcs_type, var:lp_vcs_root

Returns `true` if the current directory is part of a version control repository. If not, returns `1`. Returns the VCS type ID and the repository root directory.

If the current directory is disabled for version control using `LP_DISABLED_VCS_PATHS` (checked using `_lp_are_vcs_enabled()`), returns `2`, and the returned type is set to "disabled".

`_lp_find_vcs()` will only search for VCS types that are not disabled. If all VCS types are disabled in the config, `_lp_find_vcs()` will return `1`, as no repository will be found.

This function does a lightweight check for the existence of a version control repository, only looking for the existence of a database. It does not check if the database is valid or healthy. Use `_lp_vcs_active()` to test for that.

New in version 2.0.

`_lp_are_vcs_enabled``()`

Returns `true` if the current directory is not excluded by the config option *`LP_DISABLED_VCS_PATHS`*.

Note: All following generic functions need *`_lp_find_vcs()`* to be run first, as they need `lp_vcs_type` to be set.

`_lp_vcs_active``()`

Returns `true` if the detected VCS is enabled in Liquidprompt and the current directory is a valid repository of that type. This check should be done before running any other `_lp_vcs_*` data functions, but can be omitted for speed reasons if the checks done by *`_lp_find_vcs()`* are good enough.

New in version 2.0.

Note: All following generic functions can exit with return codes higher than their normal disabled exit code. If the active VCS does not support a feature, the data function will not be defined, and therefore the shell will return an error code much higher than `2`. Compare using greater-or-equal if checking for not supported error codes.

Unless otherwise documented, the following functions return `0` for good data, `1` for no data, and `2` or higher for unsupported function.

`_lp_vcs_bookmark``()` → var:lp_vcs_bookmark

Returns `true` if a bookmark is active in the repository. Returns the bookmark name.

Most VCS providers do not support bookmarks.

New in version 2.0.

`_lp_vcs_branch``()` → var:lp_vcs_branch

Returns `true` if a branch is active in the repository. Returns the branch name.

For some VCS providers, a branch is always active.

New in version 2.0.

`_lp_vcs_commit_id``()` → var:lp_vcs_commit_id

Returns the full commit ID of the current commit. The return code is not defined.

Some VCS providers use hashes, while others use incrementing revision numbers. All VCS providers support some form of ID. The returned string should be unique enough that a user can identify the commit.

New in version 2.0.

`_lp_vcs_commits_off_remote``()` → var:lp_vcs_commit_ahead, var:lp_vcs_commit_behind

Returns `true` if there are commits on the current branch that are not on the remote tracking branch, or commits on the remote tracking branch that are not on this branch. Returns `1` if there are no differing commits. Returns `2` if there is no matching remote tracking branch. Returns `3` or higher if the VCS provider does not support remote tracking branches.

Returns the number of commits behind and ahead.

Most VCS providers do not support remote tracking branches.

New in version 2.0.

`_lp_vcs_head_status``()` → var:lp_vcs_head_status, var:lp_vcs_head_details

Return `true` if the repo is in a special or unusual state. Return the special status, and any extra details (like progress in a rebase) if applicable.

Many VCS providers do not have such information. This info is unlikely to be similar across VCSs, and should probably be displayed to a user without manipulation.

Note: The details are optional, and might not be set. Protect it with `"${lp_vcs_head_details-}"`.

New in version 2.0.

`_lp_vcs_stash_count``()` → var:lp_vcs_stash_count
Returns `true` if there are stashes the repository. Returns the number of stashes.

Some VCS providers refer to stashes as "shelves".

Some VCS providers do not support stashes.

New in version 2.0.

`_lp_vcs_tag``()` → var:lp_vcs_tag
Returns `true` if a tag is active in the repository. Returns the tag name.

A tag will only be returned if it is a unique ID that points only to the current commit.

If multiple tags match, only one is returned. Which tag is selected is not defined.

Some VCS providers do not support unique tags.

New in version 2.0.

`_lp_vcs_uncommitted_files``()` → var:lp_vcs_uncommitted_files
Returns `true` if any uncommitted files exist in the repository. In other words, tracked files that contain uncommitted changes. Returns the number of uncommitted files.

Some VCS providers refer to uncommitted files as "modified" files.

New in version 2.0.

`_lp_vcs_uncommitted_lines``()` → var:lp_vcs_uncommitted_lines
Returns `true` if any uncommitted lines exist in the repository. In other words, tracked files that contain uncommitted changes. Returns the number of uncommitted lines.

Some VCS providers refer to uncommitted lines as "modified" or "changed" lines.

New in version 2.0.

`_lp_vcs_unstaged_files``()` → var:lp_vcs_unstaged_files
Returns `true` if any unstaged files exist in the repository. In other words, tracked files that contain unstaged changes. Returns the number of unstaged files.

Many VCS providers do not support staging.

New in version 2.0.

`_lp_vcs_unstaged_lines``()` → var:lp_vcs_unstaged_lines
Returns `true` if any unstaged lines exist in the repository. In other words, tracked files that contain unstaged changes. Returns the number of unstaged lines.

Many VCS providers do not support staging.

New in version 2.0.

`_lp_vcs_untracked_files``()` → var:lp_vcs_untracked_files
Returns `true` if any untracked files exist in the repository. Returns the number of untracked files.

Some VCS providers refer to untracked files as "extra" files.

New in version 2.0.

Bazaar

Warning: Bazaar is no longer being actively developed, and depends on Python 2, which is no longer supported. Breezy is a fork that can work with Bazaar repositories. To use Breezy in place of Bazaar, set a wrapper function:

```
bzr() { brz "$@"; }
```

Note: Bazaar does not support bookmarks. A nick is somewhat like a bookmark, but there is no command to view a naked branch name, so the `nick` command is used for branches.

Note: Bazaar does not support a staging area.

Note: Bazaar does not support getting details of remote tracking branches. Bazaar does not keep a local copy of the remote state, so checking this would be impossible anyway.

Note: Bazaar does not have extra head statuses. A Bazaar merge can be partially complete, but there is no command to test for it.

`_lp_bzr_active``()`

Returns `true` if Bazaar is enabled in Liquidprompt and the current directory is a valid Bazaar repository. This check should be done before running any other `_lp_bzr_*` data functions if accessing the Bazaar data functions directly instead of through the generic interface.

Can be disabled by *`LP_ENABLE_BZR`*.

New in version 2.0.

`_lp_bzr_branch``()` → var:lp_vcs_branch

Returns `true` if a branch is active in the repository. Returns the branch name.

Changed in version 2.0: Return method changed from stdout. No branch now returns `false`.

`_lp_bzr_commit_id``()` → var:lp_vcs_commit_id

Returns the revision number of the current commit. The return code is not defined.

New in version 2.0.

`_lp_bzr_stash_count``()` → var:lp_vcs_stash_count

Returns `true` if there are shelves the repository. Returns the number of shelves.

New in version 2.0.

`_lp_bzr_tag``()` → var:lp_vcs_tag

Returns `true` if a tag is active in the repository. Returns the tag name.

If multiple tags match, only one is returned. Which tag is selected is not defined.

New in version 2.0.

`_lp_bzr_uncommitted_files``()` → var:lp_vcs_uncommitted_files

Returns `true` if any uncommitted files exist in the repository. In other words, tracked files that contain uncommitted changes. Returns the number of uncommitted files.

New in version 2.0.

`_lp_bzr_uncommitted_lines``()` → var:lp_vcs_uncommitted_lines
Returns `true` if any uncommitted lines exist in the repository. In other words, tracked files that contain uncommitted changes. Returns the number of uncommitted lines.

New in version 2.0.

`_lp_bzr_untracked_files``()` → var:lp_vcs_untracked_files
Returns `true` if any untracked files exist in the repository. Returns the number of untracked files.

New in version 2.0.

Fossil

Note: Fossil does not support bookmarks.

Note: Fossil does not support a staging area.

Note: Fossil does not support unique tags. Fossil tags can refer to multiple checkin IDs, so a matching tag is not a useful unique ID.

Note: Fossil does not support remote tracking branches. Fossil by default keeps the local repository in sync with the remote. Even if a user disables that, it is not possible to have a local and remote branch named the same not in sync.

`_lp_fossil_active``()`
Returns `true` if Fossil is enabled in Liquidprompt and the current directory is a valid Fossil repository. This check should be done before running any other `_lp_fossil_*` data functions if accessing the Fossil data functions directly instead of through the generic interface.

Can be disabled by *`LP_ENABLE_FOSSIL`*.

New in version 2.0.

`_lp_fossil_branch``()` → var:lp_vcs_branch
Returns `true` if a branch is active in the repository. Returns the branch name.

Changed in version 2.0: Return method changed from stdout. No branch now returns `false` and nothing instead of "no-branch".

`_lp_fossil_commit_id``()` → var:lp_vcs_commit_id
Returns the full commit hash of the current commit. The return code is not defined.

New in version 2.0.

`_lp_fossil_head_status``()` → var:lp_vcs_head_status
Return `true` if the repository is in a special or unusual state. Return the special status.

Does not return any extra details.

New in version 2.0.

`_lp_fossil_stash_count``()` → var:lp_vcs_stash_count
Returns `true` if there are stashes the repository. Returns the number of stashes.

New in version 2.0.

`_lp_fossil_uncommitted_files``()` → var:lp_vcs_uncommitted_files
Returns `true` if any uncommitted files exist in the repository. In other words, tracked files that contain uncommitted changes. Returns the number of uncommitted files.

New in version 2.0.

`_lp_fossil_uncommitted_lines``()` → var:lp_vcs_uncommitted_lines
Returns `true` if any uncommitted lines exist in the repository. In other words, tracked files that contain uncommitted changes. Returns the number of uncommitted lines.

New in version 2.0.

`_lp_fossil_untracked_files``()` → var:lp_vcs_untracked_files
Returns `true` if any untracked files exist in the repository. Returns the number of untracked files.

New in version 2.0.

Git

Note: Git does not support bookmarks.

`_lp_git_active``()`
Returns `true` if Git is enabled in Liquidprompt and the current directory is a valid Git repository. This check should be done before running any other `_lp_git_*` data functions if accessing the Git data functions directly instead of through the generic interface.

Can be disabled by *`LP_ENABLE_GIT`*.

New in version 2.0.

`_lp_git_branch``()` → var:lp_vcs_branch
Returns `true` if a branch is active in the repository. Returns the branch name.

Changed in version 2.0: Return method changed from stdout. No branch now returns `false` and nothing instead of commit ID.

`_lp_git_commit_id``()` → var:lp_vcs_commit_id
Returns the full commit hash of the current commit. The return code is not defined.

New in version 2.0.

`_lp_git_commits_off_remote``()` → var:lp_vcs_commit_ahead, var:lp_vcs_commit_behind
Returns `true` if there are commits on the current branch that are not on the remote tracking branch, or commits on the remote tracking branch that are not on this branch. Returns `1` if there are no differing commits. Returns `2` if there is no matching remote tracking branch.

Returns the number of commits behind and ahead.

New in version 2.0.

`_lp_git_head_status``()` → var:lp_vcs_head_status, var:lp_vcs_head_details
Return `true` if the repository is in a special or unusual state. Return the special status, and any extra details (like progress in a rebase) if applicable.

New in version 2.0.

`_lp_git_stash_count``()` → var:lp_vcs_stash_count
Returns `true` if there are stashes the repository. Returns the number of stashes.

New in version 2.0.

`_lp_git_tag``()` → var:lp_vcs_tag

Returns `true` if a tag is active in the repository. Returns the tag name.

If multiple tags match, only one is returned. Which tag is selected is not defined.

New in version 2.0.

`_lp_git_uncommitted_files``()` → var:lp_vcs_uncommitted_files

Returns `true` if any uncommitted files exist in the repository. In other words, tracked files that contain uncommitted changes. Returns the number of uncommitted files.

New in version 2.0.

`_lp_git_uncommitted_lines``()` → var:lp_vcs_uncommitted_lines

Returns `true` if any uncommitted lines exist in the repository. In other words, tracked files that contain uncommitted changes. Returns the number of uncommitted lines.

New in version 2.0.

`_lp_git_unstaged_files``()` → var:lp_vcs_unstaged_files

Returns `true` if any unstaged files exist in the repository. In other words, tracked files that contain unstaged changes. Returns the number of unstaged files.

New in version 2.0.

`_lp_git_unstaged_lines``()` → var:lp_vcs_unstaged_lines

Returns `true` if any unstaged lines exist in the repository. In other words, tracked files that contain unstaged changes. Returns the number of unstaged lines.

New in version 2.0.

`_lp_git_untracked_files``()` → var:lp_vcs_untracked_files

Returns `true` if any untracked files exist in the repository. Returns the number of untracked files.

New in version 2.0.

Mercurial

Note: Mercurial does not support a staging area.

Note: Mercurial remote tracking branches are disabled (see *`_lp_hg_commits_off_remote()`*).

`_lp_hg_active``()`

Returns `true` if Mercurial is enabled in Liquidprompt and the current directory is a valid Mercurial repository. This check should be done before running any other `_lp_hg_*` data functions if accessing the Mercurial data functions directly instead of through the generic interface.

Can be disabled by *`LP_ENABLE_HG`*.

New in version 2.0.

`_lp_hg_bookmark``()` → var:lp_vcs_bookmark

Returns `true` if a bookmark is active in the repository. Returns the bookmark name.

Mercurial bookmarks work more like Git branches.

New in version 2.0.

_lp_hg_branch() → var:lp_vcs_branch

Returns `true` if a branch is active in the repository. Returns the branch name.

All Mercurial commits have a branch, so this function should always return `true`. A closer analog to Git branches are Mercurial bookmarks (see *_lp_hg_bookmark()*).

Changed in version 2.0: Return method changed from stdout. No branch now returns `false`.

_lp_hg_commit_id() → var:lp_vcs_commit_id

Returns the full global revision ID of the current commit. The return code is not defined.

New in version 2.0.

_lp_hg_commits_off_remote()

Returns `3` (disabled).

Mercurial does not keep a local copy of the remote state, so checking this will require a connection to the remote server. This means it is often prohibitively time expensive, and therefore should not be used in a prompt. See issue #217.

New in version 2.0.

_lp_hg_head_status() → var:lp_vcs_head_status

Return `true` if the repository is in a special or unusual state. Return the special status.

Does not return any extra details.

This function depends on *_lp_find_vcs()* being run first to set `lp_vcs_root`.

New in version 2.0.

_lp_hg_stash_count() → var:lp_vcs_stash_count

Returns `true` if there are shelves the repository. Returns the number of shelves.

New in version 2.0.

_lp_hg_tag() → var:lp_vcs_tag

Returns `true` if a tag is active in the repository. Returns the tag name.

If multiple tags match, only one is returned. Which tag is selected is not defined.

New in version 2.0.

_lp_hg_uncommitted_files() → var:lp_vcs_uncommitted_files

Returns `true` if any uncommitted files exist in the repository. In other words, tracked files that contain uncommitted changes. Returns the number of uncommitted files.

New in version 2.0.

_lp_hg_uncommitted_lines() → var:lp_vcs_uncommitted_lines

Returns `true` if any uncommitted lines exist in the repository. In other words, tracked files that contain uncommitted changes. Returns the number of uncommitted lines.

New in version 2.0.

_lp_hg_untracked_files() → var:lp_vcs_untracked_files

Returns `true` if any untracked files exist in the repository. Returns the number of untracked files.

New in version 2.0.

Subversion

Note: Subversion does not support bookmarks.

Note: Subversion does not support a staging area.

Note: Subversion does not support stashes.

Note: Subversion does not have extra head statuses. A Subversion merge is no different than a manual file change, so the repository has no extra state to track.

Note: Subversion does not support remote tracking branches (as it is not a distributed version control system).

Note: Subversion does not support tags. What are generally agreed upon as being tags are internally branches. These are returned by *`_lp_svn_branch()`*.

`_lp_svn_active``()`

Returns `true` if Subversion is enabled in Liquidprompt and the current directory is a valid Subversion repository. This check should be done before running any other `_lp_svn_*` data functions if accessing the Subversion data functions directly instead of through the generic interface.

Can be disabled by *`LP_ENABLE_SVN`*.

New in version 2.0.

`_lp_svn_branch``()` → var:lp_vcs_branch

Returns `true` if a branch is active in the repository. Returns the branch name.

Subversion "tags" are really branches under a "tag" directory. Tags are returned as their directory name, prefixed with "tag/".

Changed in version 2.0: Return method changed from stdout. No branch now returns `false` and nothing instead of the current directory.

`_lp_svn_commit_id``()` → var:lp_vcs_commit_id

Returns the revision number of the current commit. The return code is not defined.

New in version 2.0.

`_lp_svn_uncommitted_files``()` → var:lp_vcs_uncommitted_files

Returns `true` if any uncommitted files exist in the repository. In other words, tracked files that contain uncommitted changes. Returns the number of uncommitted files.

New in version 2.0.

`_lp_svn_uncommitted_lines``()` → var:lp_vcs_uncommitted_lines

Returns `true` if any uncommitted lines exist in the repository. In other words, tracked files that contain uncommitted changes. Returns the number of uncommitted lines.

New in version 2.0.

_lp_svn_untracked_files() → var:lp_vcs_untracked_files

Returns `true` if any untracked files exist in the repository. Returns the number of untracked files.

New in version 2.0.

These functions are designed to be used by themes.

All data functions will return `true` (meaning return code `0`) when they are both enabled and have data. They will return `false` (meaning return code `1`) when they do not have data. Most will return `2` when they are disabled, either through the config or because the tool they depend on is not installed. Such disable config options will be documented. Exceptions to this rule are explicitly documented.

When a function returns `false`, any return variables are not guaranteed to be set. If running with `set -u` (or when building a theme to be distributed), guard any return variable accesses with a check of the return code, or use `${var-}` syntax.

4.2.2 Battery

_lp_battery() → var:lp_battery

Returns a return code depending on the status of the battery:

- `5` if the battery feature is disabled or not available
- `4` if no battery level is found
- `3` if charging and the level is above the threshold
- `2` if charging and the level is under the threshold
- `1` if discharging and the level is above the threshold
- `0` if discharging and the level is under the threshold

Returns an integer percentage of the current battery level.

If the threshold is not surpassed, the battery level is still returned.

The threshold is configured with *LP_BATTERY_THRESHOLD*.

Can be disabled by *LP_ENABLE_BATT*.

4.2.3 Development Environment

_lp_python_env() → var:lp_python_env

Retuns `true` if a Python or Conda environment is detected. Returns the virtual environment name.

If the environment name contains a forward slash (`/`), only the substring after the last forward slash is returned.

Can be disabled by *LP_ENABLE_VIRTUALENV*.

New in version 2.0.

_lp_software_collections() → var:lp_software_collections

Returns `true` if a Red Hat Software Collection is enabled. Returns the software collection name.

If the software collection name has trailing whitespace, it is removed.

Can be disabled by *LP_ENABLE_SCLS*.

New in version 2.0.

4.2.4 Environment

`_lp_connected_display``()`
Returns `true` if there is a connected X11 display.

New in version 2.0.

`_lp_connection``()` → var:lp_connection
Returns a string matching the connection context of the shell. Valid values:

- `ssh` - connected over OpenSSH
- `lcl` - running in a local terminal
- `su` - running in a `su` or `sudo` shell
- `tel` - connected over Telnet

It is not possible for more than one context to be returned. The contexts are checked in the order listed above, and the first one found will be returned.

It is not possible for no context to be returned.

Changed in version 2.0: Return method changed from stdout.

`_lp_dirstack``()` → var:lp_dirstack
Returns `true` if directory stack support is enabled and the directory stack contains more than one directory. In that case, the return variable is set to the number of directories on the stack.

Can be enabled by *`LP_ENABLE_DIRSTACK`*.

New in version 2.0.

`_lp_error``()` → var:lp_error
Returns `true` if the last user shell command returned a non-zero exit code. Returns (in the return variable) the exit code of that command.

Can be disabled by *`LP_ENABLE_ERROR`*.

Note: The return variable `lp_error` will always be set with the last command return code, as it must be the first thing done by the prompt. This function should still be used, as it checks for the feature being disabled by the user.

New in version 2.0.

`_lp_http_proxy``()` → var:lp_http_proxy
Returns `true` if an HTTP or HTTPS proxy is enabled through environment variables in the shell. Returns the first found proxy string.

Can be disabled by *`LP_ENABLE_PROXY`*.

New in version 2.0.

`_lp_multiplexer``()` → var:lp_mulitplexer
Returns `true` if the current shell context is inside a terminal multiplexer. Returns a string matching the multiplexer:

- `tmux`
- `screen`

New in version 2.0.

4.2.5 Jobs

`_lp_detached_sessions``()` → var:lp_detached_sessions

Returns `true` if any detached multiplexer sessions are found. Returns an integer count of how many sessions were found.

Can be disabled by *`LP_ENABLE_DETACHED_SESSIONS`*.

New in version 2.0.

`_lp_jobcount``()` → var:lp_running_jobs, var:lp_stopped_jobs

Returns `true` if any shell background jobs are found. Returns an integer count of how many jobs are running and how many are stopped.

Stopped jobs are jobs suspended with Ctrl-Z.

Running jobs are jobs started with the `command &` syntax, or stopped jobs started again with the `bg` command.

Can be disabled by *`LP_ENABLE_JOBS`*.

New in version 2.0.

4.2.6 Load

`_lp_cpu_load``()` → var:lp_cpu_load

Returns a string of the system load average smallest increment, usually 5 minutes. The return code is not defined.

`_lp_load``()` → var:lp_load

Returns `true` if the system load average in centiload is greater than the threshold. Returns the system load average, divided by the number of CPU cores, in centiload units. In other words, the load average is multiplied by 100, then divided by the number of CPU cores.

If the threshold is not surpassed, the load average is still returned.

The threshold is configured with *`LP_LOAD_THRESHOLD`*.

Can be disabled by *`LP_ENABLE_LOAD`*.

New in version 2.0.

4.2.7 OS

`_lp_chroot``()` → var:lp_chroot

Returns `true` if a chroot environment is detected. Returns a string matching the chroot string if one is found.

New in version 2.0.

`_lp_hostname``()` → var:lp_hostname

Returns `true` if a hostname should be displayed. Returns `1` if the connection type is local and *`LP_HOSTNAME_ALWAYS`* is not `1`.

Returns the hostname string.

Note: The returned string is not the real hostname, instead it is the shell escape code for hostname, so the shell will substitute the real user ID when it evaluates `PS1`.

Deprecated since version 2.0: Also sets `LP_HOST_SYMBOL` to the same return string.

Can be disabled by *`LP_HOSTNAME_ALWAYS`* set to `-1`.

New in version 2.0.

_lp_sudo_active()
Returns `true` if `sudo` is currently activated with valid credentials. If `sudo` is configured to always allow a user to run commands with no password, this will always return `true`.

Can be disabled by *LP_ENABLE_SUDO*.

New in version 2.0.

_lp_user()
Returns a return code depending on the logged in user:

- `2` - the user is root
- `1` - the user is a user other than the original login user
- `0` - the user is the login user

New in version 2.0.

_lp_username() → var:lp_username
Returns `true` if a username should be displayed. Returns `1` if the user is the login user and *LP_USER_ALWAYS* is not `1`.

Returns the current user ID.

Note: The returned string is not a real user ID, instead it is the shell escape code for user, so the shell will substitute the real user ID when it evaluates `PS1`.

Can be disabled by *LP_USER_ALWAYS* set to `-1`.

New in version 2.0.

4.2.8 Runtime

_lp_runtime_format() → var:lp_runtime_format
Returns `true` if the last command runtime was greater than the threshold. Returns a formatted string of the total runtime, split into days, hours, minutes, and seconds. Ex: `3h27m6s`.

The threshold is configured with *LP_RUNTIME_THRESHOLD*.

Can be disabled by *LP_ENABLE_RUNTIME*.

New in version 2.0.

4.2.9 Temperature

_lp_temperature() → var:lp_temperature
Returns `true` if the highest system temperature is greater than the threshold. Returns the highest temperature integer.

If the threshold is not surpassed, the highest temperature is still returned.

The threshold is configured with *LP_TEMP_THRESHOLD*.

Can be disabled by *LP_ENABLE_TEMP*.

New in version 2.0: Note that a function by this name was renamed to `_lp_temperature_color`.

4.2.10 Time

`_lp_time``()` → var:lp_time

Returns `true` if digital time is enabled. Returns the current digital time string.

Note: The returned string is not the real time, instead it is the shell escape code for time, so the shell will substitute the real current time when it evaluates `PS1`.

Can be disabled by *`LP_ENABLE_TIME`* or *`LP_TIME_ANALOG`* set to `1`.

New in version 2.0.

`_lp_analog_time``()` → var:lp_analog_time

Returns `true` if analog time is enabled. Returns the current analog time as a single Unicode character, accurate to the closest 30 minutes.

Can be disabled by *`LP_ENABLE_TIME`* or *`LP_TIME_ANALOG`* set to `0`.

New in version 2.0.

4.3 Default Theme Functions

- *Theme Functions*
- *Theme Data Functions*

These functions are designed to be used by the default theme, but are documented here so that other themes can use these functions to reduce duplication if sections from the default theme are wanted.

4.3.1 Theme Functions

`_lp_default_theme_activate``()`

Setup the defaults and static pieces of the default theme.

Uses colors:

- *`LP_COLOR_IN_MULTIPLEXER`*
- *`LP_COLOR_MARK`*
- *`LP_COLOR_MARK_ROOT`*
- *`LP_COLOR_PATH_ROOT`*
- *`LP_COLOR_USER_ALT`*
- *`LP_COLOR_USER_LOGGED`*
- *`LP_COLOR_USER_ROOT`*

And marks:

- *`LP_MARK_BRACKET_OPEN`*
- *`LP_MARK_BRACKET_CLOSE`*
- *`LP_MARK_DEFAULT`*

New in version 2.0.

`_lp_default_theme_directory``()`

Setup the colors for the directory when the current working directory changes.

Uses colors:

- *`LP_COLOR_NOWRITE`*
- *`LP_COLOR_PATH`*
- *`LP_COLOR_WRITE`*

And mark *`LP_MARK_PERM`*.

New in version 2.0.

`_lp_default_theme_prompt``()`

Runs all of the below theme data functions, and writes values to the *Default Theme* variables.

If *`LP_PS1_FILE`* is set, sources it.

Then, if *`LP_PS1`* is set, uses it as `PS1`. Otherwise, uses the default theme layout to construct `PS1`.

New in version 2.0.

4.3.2 Theme Data Functions

These functions wrap *Data Functions* with color and/or other formatting. Their return codes are the same as the data functions they wrap unless otherwise documented.

The interface of the functions will not change between minor versions, but the specific text and formatting may change.

`_lp_analog_time_color``()` → var:lp_analog_time_color

Returns *`_lp_analog_time()`* with color from *`LP_COLOR_TIME`*.

New in version 2.0.

`_lp_battery_color``()` → var:lp_battery_color

Returns data from *`_lp_battery()`*, colored with:

- *`LP_COLOR_CHARGING_ABOVE`*
- *`LP_COLOR_CHARGING_UNDER`*
- *`LP_COLOR_DISCHARGING_ABOVE`*
- *`LP_COLOR_DISCHARGING_UNDER`*
- *`LP_COLORMAP`*

And using marks:

- *`LP_MARK_ADAPTER`*
- *`LP_MARK_BATTERY`*

Adds battery value if *`LP_PERCENTS_ALWAYS`* is `1`.

Changed in version 2.0: Return code matches data function. Return method changed from stdout.

`_lp_dirstack_color``()` → var:lp_dirstack_color

Returns *`_lp_dirstack()`*, prefixed with *`LP_MARK_DIRSTACK`*, all colored with *`LP_COLOR_DIRSTACK`*.

New in version 2.0.

_lp_error_color() → var:lp_error_color
Returns *_lp_error()* with color from *LP_COLOR_ERR*.

New in version 2.0.

_lp_hostname_color() → var:lp_hostname_color
Returns *_lp_hostname()*, with added data from *_lp_chroot()*. Color from *LP_COLOR_HOST*, *LP_COLOR_SSH*, `LP_COLOR_HOST_HASH`, and *LP_COLOR_TELNET*, depending on the output of *_lp_connection()*.

Added color from *_lp_connected_display()*: either *LP_COLOR_X11_ON* or *LP_COLOR_X11_OFF*.

Return code is *_lp_hostname()* ORed with *_lp_chroot()*: both must return no data for *_lp_hostname_color()* to return no data.

New in version 2.0.

_lp_http_proxy_color() → var:lp_http_proxy_color
Returns *_lp_http_proxy()* with color from *LP_COLOR_PROXY*.

New in version 2.0.

_lp_jobcount_color() → var:lp_jobcount_color
Returns *_lp_detached_sessions()* with color from *LP_COLOR_JOB_D* and *_lp_jobcount()* with colors from *LP_COLOR_JOB_R* and *LP_COLOR_JOB_Z*.

Return code is *_lp_detached_sessions()* ORed with *_lp_jobcount()*: both must return no data for *_lp_jobcount_color()* to return no data.

Changed in version 2.0: Return code matches data function. Return method changed from stdout.

_lp_load_color() → var:lp_load_color
Returns *_lp_load()* with color from *LP_COLORMAP* and mark from *LP_MARK_LOAD*.

Adds load value if *LP_PERCENTS_ALWAYS* is `1`.

Changed in version 2.0: Return code matches data function. Return method changed from stdout.

_lp_python_env_color() → var:lp_python_env_color
Returns *_lp_python_env()* with color from *LP_COLOR_VIRTUALENV*.

New in version 2.0.

_lp_runtime_color() → var:lp_runtime_color
Returns *_lp_runtime_format()* with color from *LP_COLOR_RUNTIME*.

Changed in version 2.0: Renamed from `_lp_runtime`. Return code matches data function. Return method changed from stdout.

_lp_software_collections_color() → var:lp_software_collections_color
Returns *_lp_software_collections()* with color from *LP_COLOR_VIRTUALENV*.

New in version 2.0.

_lp_sudo_active_color() → var:lp_sudo_active_color
Returns *_lp_sudo_active()* with color and marks from *LP_COLOR_MARK_SUDO* if sudo is active, or `LP_COLOR_MARK_NO_SUDO` if not.

Does not return `1` if sudo is not active, as the return string is still needed.

Changed in version 2.0: Renamed from `_lp_sudo_check`. Always defined instead of only when *LP_ENABLE_SUDO* is enabled. Return variable changed from `LP_COLOR_MARK`.

_lp_temperature_color() → var:lp_temperature_color
Returns *_lp_temperature()* with color from *LP_COLORMAP* and mark from *LP_MARK_TEMP*.

Changed in version 2.0: Renamed from `_lp_temperature`. Return code matches data function. Return method changed from stdout.

`_lp_time_color``()` → var:lp_time_color
Returns *`_lp_time()`* with color from *`LP_COLOR_TIME`*.

New in version 2.0.

`_lp_vcs_details_color``()` → var:lp_vcs_details_color
Returns data from all generic *Version Control Data Functions*, colored with:

- *`LP_COLOR_CHANGES`*
- *`LP_COLOR_COMMITS`*
- *`LP_COLOR_COMMITS_BEHIND`*
- *`LP_COLOR_DIFF`*
- *`LP_COLOR_UP`*

And using marks:

- *`LP_MARK_STASH`*
- *`LP_MARK_UNTRACKED`*

This function should only be called when in a VCS repository; use *`_lp_find_vcs()`* or *`_lp_vcs_active()`* before.

The return code is undefined; a string should always be returned.

New in version 2.0.

4.4 Utility Functions

These functions are designed to be used by themes.

`_lp_as_text``(`*string*`)` → stdout
Return *string* with all shell escaped substrings removed.

`_lp_bool``(`*variable*[, *code*]`)`
Deprecated since version 2.0.

Stores the *code* in a variable named *variable*. If *code* is not set, uses the last return code instead.

`_lp_color_map``(`*value*, *scale=100*`)` → var:ret
Returns a color from the configured or default color map based on *value* and optional *scale*.

Values below 0 or above *scale* will be capped.

The returned string is a fully escaped terminal formatting sequence.

`_lp_sb``(`*string*`)` → stdout
Deprecated since version 2.0: Use the return code of the source data function to determine if any string was returned.

If *string* is set and not empty, returns *string* padded with an extra space on the right and the left.

`_lp_sl``(`*string*`)` → stdout
Deprecated since version 2.0: Use the return code of the source data function to determine if any string was returned.

If *string* is set and not empty, returns *string* padded with an extra space on the left.

`_lp_sr` (*string*) → stdout
: Deprecated since version 2.0: Use the return code of the source data function to determine if any string was returned.

  If *string* is set and not empty, returns *string* padded with an extra space on the right.

`_lp_smart_mark` ([*vcs_type*]) → var:lp_smart_mark
: Returns a string set to the configured mark matching *vcs_type*. If *vcs_type* is not set, uses the value of `lp_vcs_type` instead.

  If the type is "git", extra checks are done to see if the repository is of type VCSH or git-svn instead, and return their marks if so.

`_lp_title` (*title*) → stdout
: Deprecated since version 2.0: Use *`_lp_formatted_title`* instead.

  Formats *title* with title escape codes. The input is escaped using *`_lp_as_text()`* to strip terminal formatting from being added to the title. The output should be added to `PS1` to be printed as a title.

  This function will do nothing if *`LP_ENABLE_TITLE`* is disabled.

`_lp_formatted_title` (*title*)
: Sets the theme generated title to *title*. The input is escaped using *`_lp_as_text()`* to strip terminal formatting from being added to the title.

  This function will do nothing and return `2` if *`LP_ENABLE_TITLE`* is disabled.

  New in version 2.0.

`_lp_raw_title` (*title*)
: Sets the theme generated title to *title*. The input is not escaped in any way: if the input contains terminal formatting, use *`_lp_title()`* instead.

  This function will do nothing and return `2` if *`LP_ENABLE_TITLE`* is disabled.

  New in version 2.0.

4.5 Internal Functions

- *Config*
- *Formatting*
- *Load*
- *Path*
- *Prompt*
- *Runtime*
- *Theme*
- *Temperature*
- *Utility*

These functions are designed to be used only by Liquidprompt internals and data functions. These functions should not be used by users or themes, as they are not guaranteed to be stable between versions. There are documented here for information for those developing Liquidprompt.

4.5.1 Config

`__lp_source_config``(`[*--no-config*]`)`
Load the user config and default config values. This function is called by *`lp_activate()`*.

Also setup color variables that can be used by the user for their color config. Those variables are local to this function.

If the `--no-config` flag is passed, defaults are set, but no config file is sourced.

Changed in version 2.0: Renamed from `_lp_source_config`. Added `--no-config` flag.

4.5.2 Formatting

`__lp_background_color``(`*color*`)` → var:ab_color
Returns the terminal escape code to set the background color to the ANSI escape color code integer *color*. No checking is done for invalid color codes.

New in version 1.12.

Changed in version 2.0: Renamed from `background_color`.

`__lp_foreground_color``(`*color*`)` → var:af_color
Returns the terminal escape code to set the foreground color to the ANSI escape color code integer *color*. No checking is done for invalid color codes.

New in version 1.12.

Changed in version 2.0: Renamed from `foreground_color`.

4.5.3 Load

`__lp_cpu_count``()` → var:_lp_CPUNUM
Returns the number of CPUs on the machine. The implementation depends on the operating system.

New in version 2.0.

4.5.4 Path

`__lp_path``()` → var:lp_path
Sets `lp_path` to the current path. Internally, calls either *`__lp_shorten_path()`* or *`__lp_set_dirtrim()`* to do so, depending on the values of *`LP_PATH_KEEP`* and *`LP_ENABLE_SHORTEN_PATH`*.

Only *`__lp_shorten_path()`* actually sets `lp_path`, other config options will initialize `lp_path` with a shell escape sequence so the shell will print the path.

New in version 2.0.

`__lp_pwd_tilde``()` → var:lp_pwd_tilde
Returns `PWD` with the user's home directory replace with a tilde ("~").

Changed in version 2.0: Renamed from `_lp_get_home_tilde_collapsed`. Return method changed from stdout.

`__lp_set_dirtrim``()` → var:PROMPT_DIRTRIM
In Bash shells, `PROMPT_DIRTRIM` is the number of directories to keep at the end of the displayed path (if

"w" is present in `PS1`). Liquid Prompt can calculate this number under two conditions, path shortening must be disabled and `PROMPT_DIRTRIM` must be already set.

Changed in version 2.0: Renamed from `_lp_set_dirtrim`.

`__lp_shorten_path``()` → var:lp_shorten_path

Shorten the path of the current working directory if the path is longer than *`LP_PATH_LENGTH`*. Show as much of the current working directory path as possible. If shortened display a leading mark, such as ellipses, to indicate that part is missing. Show at least *`LP_PATH_KEEP`* leading directories and current directory.

Changed in version 2.0: Renamed from `_lp_shorten_path`. Removed handling of cases where no shortening is required, as that should be handled by *`__lp_path`* on activate. Return variable changed from `LP_PWD`.

4.5.5 Prompt

`__lp_set_prompt``()`

Setup features that need to be handled outside of the themes, like *`_lp_error()`* (since last return code must be recorded first), non printing features like *`LP_ENABLE_RUNTIME_BELL`* and *`LP_ENABLE_TITLE`*, and track current directory changes. This function also calls the current theme functions.

Changed in version 2.0: Renamed from `_lp_set_prompt`.

4.5.6 Runtime

`__lp_runtime_before``()`

Hooks into the shell to run directly after the user hits return on a command, to record the current time before the command runs.

Changed in version 2.0: Renamed from `_lp_runtime_before`.

`__lp_runtime_after``()`

Hooks into the shell to run directly after the user command returns, to record the current time, and calculate how long the command ran for.

Changed in version 2.0: Renamed from `_lp_runtime_after`.

4.5.7 Theme

`__lp_theme_list``()` → var:lp_theme_list

Returns an array of Liquidprompt themes currently loaded in memory. Looks for functions matching `_lp_*_theme_prompt`.

New in version 2.0.

`__lp_theme_bash_complete``()` → var:COMPREPLY

Uses *`__lp_theme_list()`* to provide Bash autocompletion for *`lp_theme()`*.

New in version 2.0.

`__lp_theme_zsh_complete``()`

Uses *`__lp_theme_list()`* to provide Zsh autocompletion for *`lp_theme()`*.

New in version 2.0.

4.5.8 Temperature

`__lp_temp_detect``()` → var:_LP_TEMP_FUNCTION

Attempts to run the possible temperature backend functions below to find one that works correctly. When one correctly returns a value, it is saved to `_LP_TEMP_FUNCTION` for use by *`_lp_temperature()`*.

Changed in version 2.0: Renamed from `_lp_temp_detect`.

`__lp_temp_acpi``()` → var:lp_temperature

A temperature backend using `acpi`.

Changed in version 2.0: Renamed from `_lp_temp_acpi`. Return variable changed from `temperature`.

`__lp_temp_sensors``()` → var:lp_temperature

A temperature backend using lm-sensors provided `sensors`.

Changed in version 2.0: Renamed from `_lp_temp_sensors`. Return variable changed from `temperature`.

4.5.9 Utility

`__lp_escape``(`*string*`)` → var:ret

Escape shell escape characters so they print correctly in `PS1`.

In Bash, backslashes (`\`) are used to escape codes, so backslashes are replaced by two backslashes.

In Zsh, percents (`%`) are used to escape codes, so percents are replaced by two percents.

Changed in version 2.0: Renamed from `_lp_escape`. Return method changed from stdout.

`__lp_is_function``(`*function*`)`

Returns `true` if *function* is the name of a function.

New in version 2.0.

`__lp_line_count``(`*string*`)` → var:count

Count the number of newline characters (`\n`) in *string*. A faster drop-in replacement for `wc -l`.

New in version 2.0.

CHAPTER

FIVE

RELEASE NOTES

The release documents here are a brief overview of the most important changes in a release. See the Changelog for a full description of what has changed.

For a full description of what a user needs to do to upgrade, see *Upgrading Liquidprompt*.

5.1 Version 2.0 Release Notes

Version 2.0 had to break a few eggs, but got a lot of reward out of it. This release is full of new features and improvements.

- *Speed Improvements*
- *Theme Engine*
- *Example Themes*
- *Data Sources*
- *Version Control Interface*
- *Version Control Tracking updates without directory change*
- *Activate Function*
- *Documentation*

5.1.1 Speed Improvements

By removing subshells, `exec`, and other `fork` ing calls, the whole project has seen incredible speed improvements, anywhere from 1.5 to 10 times as fast.

5.1.2 Theme Engine

Thanks to the new data source functions (see below), themes are able to change *everything* about how the prompt is displayed, instead of only color and element order. See *Theming*.

5.1.3 Example Themes

Liquidprompt now ships with some example themes showcasing how the new theme engine works. They are also fulling working themes that you can use as your daily drivers. See *Included Themes*.

5.1.4 Data Sources

To power the themes, all of the data sources in Liquidprompt have been broken out into individual data functions that can be called by themes. They are also documented in detail in *Data Functions*.

5.1.5 Version Control Interface

Before, each version control provider had its own function for displaying repository information. Now there is a unifying interface over all VCS providers that themes can use to display any VCS provider the same as all the others. See *Version Control Data Functions*.

The default theme now uses this interface to display all VCS providers in the same way (similar to how Git was displayed before). See *Default Theme Functions*.

5.1.6 Version Control Tracking updates without directory change

Before, if `git init` or similar was run in a directory, Liquidprompt would not display any repository information until the current directory was changed. Thanks to the speed improvements, Liquidprompt now checks for a repository at each prompt, while still being faster than version 1.12.

5.1.7 Activate Function

Before, when changing the user config file, a user needed to source `liquidprompt` again to load their config changes (or `exec bash` or `exec zsh`). Now that all of the initialization code has been refactored into *`lp_activate`*, running *`lp_activate`* after modifying the config file or installing a new feature dependent program like `git` is all that is needed!

5.1.8 Documentation

The often lacking README documentation has been re-written with Sphinx to make this much improved documentation source.

5.2 Version 1.12 Release Notes

Most of the changes in 1.12 are accumulated bug fixes, but a few features made it in as well.

- *Runtime Bell*
- *Permissions Mark*
- *Preset Color Aliases*
- *Speed Improvements*

5.2.1 Runtime Bell

A new feature, similar to the displayed last command runtime, is to ring the terminal bell when the running command exits, if the runtime was over a threshold. This can be used to notify when a long running command has finished.

See the `LP_RUNTIME_BELL` and *`LP_RUNTIME_BELL_THRESHOLD`* config options.

5.2.2 Permissions Mark

The `:` mark between the hostname and the current directory was a constant string, but now it has a config option: *`LP_MARK_PERM`*.

5.2.3 Preset Color Aliases

The `5` value of the basic colors is often named “magenta”, but in Liquidprompt it has always been “purple”, and the bold version is “pink”.

To make the options more standard, an alias for `PURPLE` is `MAGENTA`, and `PINK` now has aliases of `BOLD_PURPLE` and `BOLD_MAGENTA`.

5.2.4 Speed Improvements

Improvements to the startup process have cut startup times by at least 30% in all cases.

CHAPTER

SIX

UPGRADING LIQUIDPROMPT

We try our best to make new versions of Liquidprompt backwards compatible with previous versions, but sometimes things need to change to make forward progress. If a version introduces breaking changes or deprecation notices, a detailed document describing what changes a user needs to make will be linked below.

6.1 Version 2.0 Upgrade Notes

Most of the changes in 2.0 are in private functions and variables. There are three public API changes that could impact users: *$lp_err*, *$LP_DISABLED_VCS_PATH*, and *_lp_title()*. The rest are private API changes, but are still documented here.

- *Breaking Changes*
  - *Public Breaking Changes*
    - *$lp_err*
  - *Private Breaking Changes*
    - *$_LP_SHELL_bash*
    - *$_LP_SHELL_zsh*
    - *_lp_battery()*
    - *_lp_battery_color()*
    - *_lp_bzr_branch()*
    - *_lp_bzr_branch_color()*
    - *_lp_color_map()*
    - *_lp_connection()*
    - *_lp_escape()*
    - *_lp_fossil_branch()*
    - *_lp_fossil_branch_color()*
    - *_lp_get_home_tilde_collapsed()*
    - *_lp_git_branch()*
    - *_lp_git_branch_color()*

    * *_lp_git_head_status()*
    * *_lp_hg_branch()*
    * *_lp_hg_branch_color()*
    * *_lp_jobcount_color()*
    * *_lp_load_color()*
    * *_lp_runtime()*
    * *_lp_runtime_after()*
    * *_lp_runtime_before()*
    * *_lp_set_dirtrim()*
    * *_lp_set_prompt()*
    * *_lp_shorten_path()*
    * *_lp_smart_mark()*
    * *_lp_source_config()*
    * *_lp_svn_branch()*
    * *_lp_svn_branch_color()*
    * *_lp_temp_acpi()*
    * *_lp_temp_detect()*
    * *_lp_temp_sensors()*
    * *_lp_temperature()*
    * *_lp_time()*
    * *_lp_upwards_find()*
- *Deprecations*
  - *Public Deprecations*
    * *$LP_DISABLED_VCS_PATH*
    * *_lp_title()*
  - *Private Deprecations*
    * *_lp_bool()*
    * *_lp_sb()*
    * *_lp_sl()*
    * *_lp_sr()*

6.1.1 Breaking Changes

Public Breaking Changes

$lp_err

Renamed to `$lp_error`. Instead of referencing it directly, use *`_lp_error()`*.

Private Breaking Changes

$_LP_SHELL_bash

Now returns `1` or `0` instead of `true` or `false`

Replace test statements like:

```
if $_LP_SHELL_bash; then
```

with:

```
if (( $_LP_SHELL_bash )); then
```

$_LP_SHELL_zsh

Now returns `1` or `0` instead of `true` or `false`

Replace test statements like:

```
if $_LP_SHELL_zsh; then
```

with:

```
if (( $_LP_SHELL_zsh )); then
```

_lp_battery()

Return changed from stdout to `$lp_battery`

Replace assignment statements like:

```
battery="$(_lp_battery)"
```

with:

```
local lp_battery
_lp_battery
battery=$lp_battery
```

See also: *`_lp_battery()`*.

_lp_battery_color()

Return changed from stdout to `$lp_battery_color`

Replace assignment statements like:

```
battery_color="$(_lp_battery_color)"
```

with:

```
local lp_battery_color
_lp_battery_color
battery_color=$lp_battery_color
```

See also: *_lp_battery_color()*.

_lp_bzr_branch()

Return changed from stdout to `$lp_vcs_branch`

Recommended that *_lp_vcs_branch()* is used instead.

Replace assignment statements like:

```
branch="$(_lp_bzr_branch)"
```

with:

```
local lp_vcs_branch
if _lp_bzr_branch; then
    branch=$lp_vcs_branch
fi
```

_lp_bzr_branch_color()

Removed, replace by *_lp_vcs_details_color()*.

If the exact previous output is needed, you can implement a theme function using *Version Control Data Functions*.

Replace assignment statements like:

```
LP_VCS="$(_lp_bzr_branch_color)"
```

with:

```
if _lp_find_vcs;
    local lp_vcs_details_color
    _lp_vcs_details_color
    LP_VCS=$lp_vcs_details_color
fi
```

_lp_color_map()

Return changed from stdout to `$ret`

Replace assignment statements like:

```
output="$(_lp_color_map "$input")"
```

with:

```
local ret
_lp_color_map "$input"
output=$ret
```

See also: *_lp_color_map()*.

_lp_connection()

Return changed from stdout to `$lp_connection`

Replace assignment statements like:

```
connection="$(_lp_connection)"
```

with:

```
local lp_connection
_lp_connection
connection=$lp_connection
```

See also: *_lp_connection()*.

_lp_escape()

Renamed to *__lp_escape*. Return changed from stdout to `$ret`

Replace assignment statements like:

```
output="$(_lp_escape "$input")"
```

with:

```
local ret
__lp_escape "$input"
output=$ret
```

_lp_fossil_branch()

Return changed from stdout to `$lp_vcs_branch`

Recommended that *`_lp_vcs_branch()`* is used instead.

No longer returns "no-branch" if branch not found.

Replace assignment statements like:

```
branch="$(_lp_fossil_branch)"
```

with:

```
local lp_vcs_branch
if _lp_fossil_branch; then
    branch=$lp_vcs_branch
else
    branch="no-branch"
fi
```

_lp_fossil_branch_color()

Removed, replace by *`_lp_vcs_details_color()`*.

If the exact previous output is needed, you can implement a theme function using *Version Control Data Functions*.

Replace assignment statements like:

```
LP_VCS="$(_lp_fossil_branch_color)"
```

with:

```
if _lp_find_vcs;
    local lp_vcs_details_color
    _lp_vcs_details_color
    LP_VCS=$lp_vcs_details_color
fi
```

_lp_get_home_tilde_collapsed()

Renamed to *`__lp_pwd_tilde()`*.

Return changed from stdout to `$lp_pwd_tilde`

Recommended that `lp_path` is used instead.

Replace assignment statements like:

```
working_dir="$(_lp_get_home_tilde_collapsed)"
```

with:

```
local lp_pwd_tilde
__lp_pwd_tilde
working_dir=$lp_pwd_tilde
```

_lp_git_branch()

Return changed from stdout to `$lp_vcs_branch`

Recommended that *_lp_vcs_branch()* is used instead.

No longer returns commit hash if branch not found.

Replace assignment statements like:

```
branch="$(_lp_git_branch)"
```

with:

```
local lp_vcs_branch
if _lp_git_branch; then
    branch=$lp_vcs_branch
else
    local lp_vcs_commit_id
    _lp_git_commit_id
    branch=$lp_vcs_commit_id
fi
```

_lp_git_branch_color()

Removed, replace by *_lp_vcs_details_color()*.

Replace assignment statements like:

```
LP_VCS="$(_lp_git_branch_color)"
```

with:

```
if _lp_find_vcs;
    local lp_vcs_details_color
    _lp_vcs_details_color
    LP_VCS=$lp_vcs_details_color
fi
```

_lp_git_head_status()

Return changed from stdout to `$lp_vcs_head_status`

Recommended that *_lp_vcs_head_status()* is used instead.

Replace assignment statements like:

```
head_status="$(_lp_git_head_status)"
```

with:

```
local lp_vcs_head_status
_lp_git_head_status
head_status=$lp_vcs_head_status
```

See also: *_lp_git_head_status()*.

_lp_hg_branch()

Return changed from stdout to `$lp_vcs_branch`

Recommended that *`_lp_vcs_branch()`* is used instead.

Replace assignment statements like:

```
branch="$(_lp_hg_branch)"
```

with:

```
local lp_vcs_branch
if _lp_hg_branch; then
    branch=$lp_vcs_branch
fi
```

_lp_hg_branch_color()

Removed, replace by *`_lp_vcs_details_color()`*.

If the exact previous output is needed, you can implement a theme function using *Version Control Data Functions*.

Replace assignment statements like:

```
LP_VCS="$(_lp_hg_branch_color)"
```

with:

```
if _lp_find_vcs;
    local lp_vcs_details_color
    _lp_vcs_details_color
    LP_VCS=$lp_vcs_details_color
fi
```

_lp_jobcount_color()

Return changed from stdout to `$lp_jobcount_color`

Replace assignment statements like:

```
jobcount_color="$(_lp_jobcount_color)"
```

with:

```
local lp_jobcount_color
_lp_jobcount_color
jobcount_color=$lp_jobcount_color
```

See also: *`_lp_jobcount_color()`*.

_lp_load_color()

Return changed from stdout to `$lp_load_color`

Replace assignment statements like:

```
load_color="$(_lp_load_color)"
```

with:

```
local lp_load_color
_lp_load_color
load_color=$lp_load_color
```

See also: *_lp_load_color()*.

_lp_runtime()

Renamed to *_lp_runtime_color()*.

Return changed from stdout to `$lp_runtime_color`

Replace assignment statements like:

```
runtime_color="$(_lp_runtime)"
```

with:

```
local lp_runtime_color
_lp_runtime_color
runtime_color=$lp_runtime_color
```

_lp_runtime_after()

Renamed to *__lp_runtime_after()*.

Recommended to not use this internal function.

_lp_runtime_before()

Renamed to *__lp_runtime_before()*.

Recommended to not use this internal function.

_lp_set_dirtrim()

Renamed to *__lp_set_dirtrim()*.

Recommended that `lp_path` is used instead.

_lp_set_prompt()

Renamed to `__lp_set_prompt`.

Recommended to not use this internal function.

_lp_shorten_path()

Renamed to `__lp_shorten_path()`.

Recommended that `lp_path` is used instead.

Return changed from stdout to `$lp_shorten_path`

Replace assignment statements like:

```
cwd="$(_lp_shorten_path)"
```

with:

```
local lp_shorten_path
__lp_shorten_path
cwd=$lp_shorten_path
```

_lp_smart_mark()

Return changed from stdout to `$lp_smart_mark`

Replace assignment statements like:

```
mark="$(_lp_smart_mark)"
```

with:

```
local lp_smart_mark
_lp_smart_mark
mark=$lp_smart_mark
```

See also: `_lp_smart_mark()`.

_lp_source_config()

Renamed to `__lp_source_config`.

Recommended to not use this internal function.

_lp_svn_branch()

Return changed from stdout to `$lp_vcs_branch`

Recommended that *`_lp_vcs_branch()`* is used instead.

No longer returns directory name if branch not found.

Replace assignment statements like:

```
branch="$(_lp_svn_branch)"
```

with:

```
local lp_vcs_branch
if _lp_svn_branch; then
    branch=$lp_vcs_branch
else
    local lp_vcs_commit_id
    _lp_svn_commit_id
    branch=$lp_vcs_commit_id
fi
```

_lp_svn_branch_color()

Removed, replace by *`_lp_vcs_details_color()`*.

If the exact previous output is needed, you can implement a theme function using *Version Control Data Functions*.

Replace assignment statements like:

```
LP_VCS="$(_lp_svn_branch_color)"
```

with:

```
if _lp_find_vcs;
    local lp_vcs_details_color
    _lp_vcs_details_color
    LP_VCS=$lp_vcs_details_color
fi
```

_lp_temp_acpi()

Renamed to *`__lp_temp_acpi()`*.

Recommended that *`_lp_temperature()`* is used instead.

Return changed from `$temperature` to `$lp_temperature`.

Replace statements like:

```
_lp_temp_acpi
# use $temperature
```

with:

```
__lp_temp_acpi
# use $lp_temperature
```

_lp_temp_detect()

Renamed to `__lp_temp_detect()`.

Recommended to not use this internal function.

_lp_temp_sensors()

Renamed to `__lp_temp_sensors()`.

Recommended that `_lp_temperature()` is used instead.

Return changed from `$temperature` to `$lp_temperature`.

Replace statements like:

```
_lp_temp_sensors
# use $temperature
```

with:

```
__lp_temp_sensors
# use $lp_temperature
```

_lp_temperature()

Renamed to `_lp_temperature_color()`.

Return changed from stdout to `$lp_temperature_color`

Replace assignment statements like:

```
temp_color="$(_lp_temperature)"
```

with:

```
local lp_temperature_color
_lp_temperature_color
temp_color=$lp_temperature_color
```

Not to be confused with the new `_lp_temperature()`.

_lp_time()

Split into `_lp_time()`, `_lp_time_color()`, `_lp_analog_time()`, and *_lp_analog_time_color()*.

The return value is no longer stored in `LP_TIME`.

Replace statements like:

```
_lp_time
```

with:

```
local lp_time_color lp_analog_time_color
if _lp_time_color; then
    LP_TIME="${lp_time_color} "
elif _lp_analog_time_color; then
    LP_TIME="${lp_analog_time_color} "
else
    LP_TIME=
fi
```

_lp_upwards_find()

Replaced by `_lp_find_vcs()`.

Replace statements like:

```
_lp_upwards_find .hg || return
```

with:

```
local lp_vcs_type lp_vcs_root
_lp_find_vcs && [[ $lp_vcs_type == hg ]] || return
```

6.1.2 Deprecations

Public Deprecations

$LP_DISABLED_VCS_PATH

Replaced by `LP_DISABLED_VCS_PATHS` array variable.

Replace a set statement like:

```
LP_DISABLED_VCS_PATH="/my/one/path:/my/other/path"
```

with:

```
LP_DISABLED_VCS_PATHS=("/my/one/path" "/my/other/path")
```

_lp_title()

Replaced by *_lp_formatted_title()*.

Most likely would have been used in a template or `.ps1` file.

Replace a call like:

```
LP_TITLE="$(_lp_title "$PS1")"
PS1="${LP_TITLE}${PS1}"
```

with:

```
_lp_formatted_title "$PS1"
```

Private Deprecations

_lp_bool()

Replaced by manually storing return codes.

Most often, the return code can be used in an `if` block, and never needs to be stored:

```
if _lp_http_proxy; then
...
```

If the function returns a more complicated return code, you can store it like this:

```
_lp_user
local -i code=$?
```

or like this if the code only matters if it is not zero:

```
_lp_user || local -i code=$?
```

_lp_sb()

Replaced by data functions indicating if they returned data or not. For example:

```
if _lp_http_proxy; then
    my_data="${lp_http_proxy} "
else
    my_data=""
fi
```

If the string source is not a data function, you can replace this function with a structure like:

```
[[ -n $my_data ]] && my_data=" ${my_data} "
```

With spaces before or after as needed.

_lp_sl()

See `_lp_sb()` above.

_lp_sr()

See `_lp_sb()` above.

CHAPTER SEVEN

INDICES AND TABLES

- genindex
- search

INDEX

Symbols

__lp_background_color()
  built-in function, 53
__lp_cpu_count()
  built-in function, 53
__lp_escape()
  built-in function, 55
__lp_foreground_color()
  built-in function, 53
__lp_is_function()
  built-in function, 55
__lp_line_count()
  built-in function, 55
__lp_path()
  built-in function, 53
__lp_pwd_tilde()
  built-in function, 53
__lp_runtime_after()
  built-in function, 54
__lp_runtime_before()
  built-in function, 54
__lp_set_dirtrim()
  built-in function, 53
__lp_set_prompt()
  built-in function, 54
__lp_shorten_path()
  built-in function, 54
__lp_source_config()
  built-in function, 53
__lp_temp_acpi()
  built-in function, 55
__lp_temp_detect()
  built-in function, 55
__lp_temp_sensors()
  built-in function, 55
__lp_theme_bash_complete()
  built-in function, 54
__lp_theme_list()
  built-in function, 54
__lp_theme_zsh_complete()
  built-in function, 54
_lp_analog_time()
  built-in function, 48
_lp_analog_time_color()
  built-in function, 49
_lp_are_vcs_enabled()
  built-in function, 35
_lp_as_text()
  built-in function, 51
_lp_battery()
  built-in function, 44
_lp_battery_color()
  built-in function, 49
_lp_bool()
  built-in function, 51
_lp_bzr_active()
  built-in function, 38
_lp_bzr_branch()
  built-in function, 38
_lp_bzr_commit_id()
  built-in function, 38
_lp_bzr_stash_count()
  built-in function, 38
_lp_bzr_tag()
  built-in function, 38
_lp_bzr_uncommitted_files()
  built-in function, 38
_lp_bzr_uncommitted_lines()
  built-in function, 39
_lp_bzr_untracked_files()
  built-in function, 39
_lp_chroot()
  built-in function, 46
_lp_color_map()
  built-in function, 51
_lp_connected_display()
  built-in function, 45
_lp_connection()
  built-in function, 45
_lp_cpu_load()
  built-in function, 46
_lp_default_theme_activate()
  built-in function, 48
_lp_default_theme_directory()

built-in function, 49
_lp_default_theme_prompt()
built-in function, 49
_lp_detached_sessions()
built-in function, 46
_lp_dirstack()
built-in function, 45
_lp_dirstack_color()
built-in function, 49
_lp_error()
built-in function, 45
_lp_error_color()
built-in function, 49
_lp_find_vcs()
built-in function, 35
_lp_formatted_title()
built-in function, 52
_lp_fossil_active()
built-in function, 39
_lp_fossil_branch()
built-in function, 39
_lp_fossil_commit_id()
built-in function, 39
_lp_fossil_head_status()
built-in function, 39
_lp_fossil_stash_count()
built-in function, 39
_lp_fossil_uncommitted_files()
built-in function, 40
_lp_fossil_uncommitted_lines()
built-in function, 40
_lp_fossil_untracked_files()
built-in function, 40
_lp_git_active()
built-in function, 40
_lp_git_branch()
built-in function, 40
_lp_git_commit_id()
built-in function, 40
_lp_git_commits_off_remote()
built-in function, 40
_lp_git_head_status()
built-in function, 40
_lp_git_stash_count()
built-in function, 40
_lp_git_tag()
built-in function, 41
_lp_git_uncommitted_files()
built-in function, 41
_lp_git_uncommitted_lines()
built-in function, 41
_lp_git_unstaged_files()
built-in function, 41
_lp_git_unstaged_lines()
built-in function, 41
_lp_git_untracked_files()
built-in function, 41
_lp_hg_active()
built-in function, 41
_lp_hg_bookmark()
built-in function, 41
_lp_hg_branch()
built-in function, 41
_lp_hg_commit_id()
built-in function, 42
_lp_hg_commits_off_remote()
built-in function, 42
_lp_hg_head_status()
built-in function, 42
_lp_hg_stash_count()
built-in function, 42
_lp_hg_tag()
built-in function, 42
_lp_hg_uncommitted_files()
built-in function, 42
_lp_hg_uncommitted_lines()
built-in function, 42
_lp_hg_untracked_files()
built-in function, 42
_lp_hostname()
built-in function, 46
_lp_hostname_color()
built-in function, 50
_lp_http_proxy()
built-in function, 45
_lp_http_proxy_color()
built-in function, 50
_lp_jobcount()
built-in function, 46
_lp_jobcount_color()
built-in function, 50
_lp_load()
built-in function, 46
_lp_load_color()
built-in function, 50
_lp_multiplexer()
built-in function, 45
_lp_python_env()
built-in function, 44
_lp_python_env_color()
built-in function, 50
_lp_raw_title()
built-in function, 52
_lp_runtime_color()
built-in function, 50
_lp_runtime_format()
built-in function, 47
_lp_sb()

built-in function, 51
_lp_sl()
built-in function, 51
_lp_smart_mark()
built-in function, 52
_lp_software_collections()
built-in function, 44
_lp_software_collections_color()
built-in function, 50
_lp_sr()
built-in function, 51
_lp_sudo_active()
built-in function, 47
_lp_sudo_active_color()
built-in function, 50
_lp_svn_active()
built-in function, 43
_lp_svn_branch()
built-in function, 43
_lp_svn_commit_id()
built-in function, 43
_lp_svn_uncommitted_files()
built-in function, 43
_lp_svn_uncommitted_lines()
built-in function, 43
_lp_svn_untracked_files()
built-in function, 43
_lp_temperature()
built-in function, 47
_lp_temperature_color()
built-in function, 50
_lp_time()
built-in function, 48
_lp_time_color()
built-in function, 51
_lp_title()
built-in function, 52
_lp_user()
built-in function, 47
_lp_username()
built-in function, 47
_lp_vcs_active()
built-in function, 36
_lp_vcs_bookmark()
built-in function, 36
_lp_vcs_branch()
built-in function, 36
_lp_vcs_commit_id()
built-in function, 36
_lp_vcs_commits_off_remote()
built-in function, 36
_lp_vcs_details_color()
built-in function, 51
_lp_vcs_head_status()
built-in function, 36
_lp_vcs_stash_count()
built-in function, 37
_lp_vcs_tag()
built-in function, 37
_lp_vcs_uncommitted_files()
built-in function, 37
_lp_vcs_uncommitted_lines()
built-in function, 37
_lp_vcs_unstaged_files()
built-in function, 37
_lp_vcs_unstaged_lines()
built-in function, 37
_lp_vcs_untracked_files()
built-in function, 37

B

built-in function
__lp_background_color(), 53
__lp_cpu_count(), 53
__lp_escape(), 55
__lp_foreground_color(), 53
__lp_is_function(), 55
__lp_line_count(), 55
__lp_path(), 53
__lp_pwd_tilde(), 53
__lp_runtime_after(), 54
__lp_runtime_before(), 54
__lp_set_dirtrim(), 53
__lp_set_prompt(), 54
__lp_shorten_path(), 54
__lp_source_config(), 53
__lp_temp_acpi(), 55
__lp_temp_detect(), 55
__lp_temp_sensors(), 55
__lp_theme_bash_complete(), 54
__lp_theme_list(), 54
__lp_theme_zsh_complete(), 54
_lp_analog_time(), 48
_lp_analog_time_color(), 49
_lp_are_vcs_enabled(), 35
_lp_as_text(), 51
_lp_battery(), 44
_lp_battery_color(), 49
_lp_bool(), 51
_lp_bzr_active(), 38
_lp_bzr_branch(), 38
_lp_bzr_commit_id(), 38
_lp_bzr_stash_count(), 38
_lp_bzr_tag(), 38
_lp_bzr_uncommitted_files(), 38
_lp_bzr_uncommitted_lines(), 39
_lp_bzr_untracked_files(), 39
_lp_chroot(), 46

_lp_color_map(), 51
_lp_connected_display(), 45
_lp_connection(), 45
_lp_cpu_load(), 46
_lp_default_theme_activate(), 48
_lp_default_theme_directory(), 49
_lp_default_theme_prompt(), 49
_lp_detached_sessions(), 46
_lp_dirstack(), 45
_lp_dirstack_color(), 49
_lp_error(), 45
_lp_error_color(), 49
_lp_find_vcs(), 35
_lp_formatted_title(), 52
_lp_fossil_active(), 39
_lp_fossil_branch(), 39
_lp_fossil_commit_id(), 39
_lp_fossil_head_status(), 39
_lp_fossil_stash_count(), 39
_lp_fossil_uncommitted_files(), 40
_lp_fossil_uncommitted_lines(), 40
_lp_fossil_untracked_files(), 40
_lp_git_active(), 40
_lp_git_branch(), 40
_lp_git_commit_id(), 40
_lp_git_commits_off_remote(), 40
_lp_git_head_status(), 40
_lp_git_stash_count(), 40
_lp_git_tag(), 41
_lp_git_uncommitted_files(), 41
_lp_git_uncommitted_lines(), 41
_lp_git_unstaged_files(), 41
_lp_git_unstaged_lines(), 41
_lp_git_untracked_files(), 41
_lp_hg_active(), 41
_lp_hg_bookmark(), 41
_lp_hg_branch(), 41
_lp_hg_commit_id(), 42
_lp_hg_commits_off_remote(), 42
_lp_hg_head_status(), 42
_lp_hg_stash_count(), 42
_lp_hg_tag(), 42
_lp_hg_uncommitted_files(), 42
_lp_hg_uncommitted_lines(), 42
_lp_hg_untracked_files(), 42
_lp_hostname(), 46
_lp_hostname_color(), 50
_lp_http_proxy(), 45
_lp_http_proxy_color(), 50
_lp_jobcount(), 46
_lp_jobcount_color(), 50
_lp_load(), 46
_lp_load_color(), 50
_lp_multiplexer(), 45
_lp_python_env(), 44
_lp_python_env_color(), 50
_lp_raw_title(), 52
_lp_runtime_color(), 50
_lp_runtime_format(), 47
_lp_sb(), 51
_lp_sl(), 51
_lp_smart_mark(), 52
_lp_software_collections(), 44
_lp_software_collections_color(), 50
_lp_sr(), 51
_lp_sudo_active(), 47
_lp_sudo_active_color(), 50
_lp_svn_active(), 43
_lp_svn_branch(), 43
_lp_svn_commit_id(), 43
_lp_svn_uncommitted_files(), 43
_lp_svn_uncommitted_lines(), 43
_lp_svn_untracked_files(), 43
_lp_temperature(), 47
_lp_temperature_color(), 50
_lp_time(), 48
_lp_time_color(), 51
_lp_title(), 52
_lp_user(), 47
_lp_username(), 47
_lp_vcs_active(), 36
_lp_vcs_bookmark(), 36
_lp_vcs_branch(), 36
_lp_vcs_commit_id(), 36
_lp_vcs_commits_off_remote(), 36
_lp_vcs_details_color(), 51
_lp_vcs_head_status(), 36
_lp_vcs_stash_count(), 37
_lp_vcs_tag(), 37
_lp_vcs_uncommitted_files(), 37
_lp_vcs_uncommitted_lines(), 37
_lp_vcs_unstaged_files(), 37
_lp_vcs_unstaged_lines(), 37
_lp_vcs_untracked_files(), 37
lp_activate(), 33
lp_terminal_format(), 34
lp_theme(), 33
lp_title(), 33
prompt_OFF(), 34
prompt_off(), 34
prompt_on(), 34
prompt_tag(), 34

E

environment variable
    PROMPT_COMMAND, 6, 34
    PROMPT_DIRTRIM, 53, 54
    PS1, 20, 30, 34, 46–49, 52, 54, 55

PWD, 53
XDG_CONFIG_DIRS, 7
XDG_CONFIG_HOME, 7

L

lp_activate()
built-in function, 33
LP_BATT, 20
LP_BATTERY_THRESHOLD, 12
LP_BRACKET_CLOSE, 21
LP_BRACKET_OPEN, 21
LP_COLOR_CHANGES, 15
LP_COLOR_CHARGING_ABOVE, 15
LP_COLOR_CHARGING_UNDER, 15
LP_COLOR_COMMITS, 15
LP_COLOR_COMMITS_BEHIND, 15
LP_COLOR_DIFF, 16
LP_COLOR_DIRSTACK, 16
LP_COLOR_DISCHARGING_ABOVE, 16
LP_COLOR_DISCHARGING_UNDER, 16
LP_COLOR_ERR, 16
LP_COLOR_HOST, 16
LP_COLOR_IN_MULTIPLEXER, 16
LP_COLOR_JOB_D, 16
LP_COLOR_JOB_R, 16
LP_COLOR_JOB_Z, 16
LP_COLOR_MARK, 16
LP_COLOR_MARK_ROOT, 16
LP_COLOR_MARK_SUDO, 16
LP_COLOR_NOWRITE, 16
LP_COLOR_PATH, 16
LP_COLOR_PATH_ROOT, 17
LP_COLOR_PROXY, 17
LP_COLOR_RUNTIME, 17
LP_COLOR_SSH, 17
LP_COLOR_SU, 17
LP_COLOR_TELNET, 17
LP_COLOR_TIME, 17
LP_COLOR_UP, 17
LP_COLOR_USER_ALT, 17
LP_COLOR_USER_LOGGED, 17
LP_COLOR_USER_ROOT, 17
LP_COLOR_VIRTUALENV, 17
LP_COLOR_WRITE, 17
LP_COLOR_X11_OFF, 17
LP_COLOR_X11_ON, 17
LP_COLORMAP, 15
LP_DIRSTACK, 21
LP_DISABLED_VCS_PATH, 8
LP_DISABLED_VCS_PATHS, 8
LP_ENABLE_BATT, 8
LP_ENABLE_BZR, 9
LP_ENABLE_COLOR, 9
LP_ENABLE_DETACHED_SESSIONS, 9
LP_ENABLE_DIRSTACK, 9
LP_ENABLE_ERROR, 9
LP_ENABLE_FOSSIL, 9
LP_ENABLE_FQDN, 9
LP_ENABLE_GIT, 9
LP_ENABLE_HG, 10
LP_ENABLE_JOBS, 10
LP_ENABLE_LOAD, 10
LP_ENABLE_PERM, 10
LP_ENABLE_PROXY, 10
LP_ENABLE_RUNTIME, 10
LP_ENABLE_RUNTIME_BELL, 10
LP_ENABLE_SCLS, 10
LP_ENABLE_SCREEN_TITLE, 10
LP_ENABLE_SHORTEN_PATH, 10
LP_ENABLE_SSH_COLORS, 11
LP_ENABLE_SUDO, 11
LP_ENABLE_SVN, 11
LP_ENABLE_TEMP, 11
LP_ENABLE_TIME, 11
LP_ENABLE_TITLE, 11
LP_ENABLE_VCS_ROOT, 11
LP_ENABLE_VIRTUALENV, 11
LP_ERR, 22
LP_HOST, 21
LP_HOSTNAME_ALWAYS, 11
LP_JOBS, 21
LP_LOAD, 20
LP_LOAD_THRESHOLD, 12
LP_MARK, 22
LP_MARK_ADAPTER, 13
LP_MARK_BATTERY, 13
LP_MARK_BRACKET_CLOSE, 13
LP_MARK_BRACKET_OPEN, 13
LP_MARK_BZR, 13
LP_MARK_DEFAULT, 13
LP_MARK_DIRSTACK, 13
LP_MARK_DISABLED, 13
LP_MARK_FOSSIL, 13
LP_MARK_GIT, 13
LP_MARK_HG, 13
LP_MARK_LOAD, 13
LP_MARK_PERM, 13
LP_MARK_PREFIX, 8
LP_MARK_PROXY, 14
LP_MARK_SHORTEN_PATH, 14
LP_MARK_STASH, 14
LP_MARK_SVN, 14
LP_MARK_TEMP, 14
LP_MARK_UNTRACKED, 14
LP_MARK_VCSH, 14
LP_PATH_DEFAULT, 8
LP_PATH_KEEP, 8
LP_PATH_LENGTH, 8

LP_PERCENTS_ALWAYS, 11
LP_PERM, 21
LP_PROXY, 21
LP_PS1, 20
LP_PS1_FILE, 20
LP_PS1_POSTFIX, 8
LP_PS1_PREFIX, 8
LP_PWD, 21
LP_RUNTIME, 22
LP_RUNTIME_BELL_THRESHOLD, 12
LP_RUNTIME_THRESHOLD, 12
LP_SCLS, 21
LP_TEMP, 21
LP_TEMP_THRESHOLD, 12
lp_terminal_format()
    built-in function, 34
lp_theme()
    built-in function, 33
LP_TIME, 20
LP_TIME_ANALOG, 12
lp_title()
    built-in function, 33
LP_USER, 21
LP_USER_ALWAYS, 12
LP_VCS, 21
LP_VENV, 21

P

POWERLINE_BATTERY_COLOR, 29
POWERLINE_CHROOT_COLOR, 29
POWERLINE_CHROOT_MARKER, 28
POWERLINE_DIRSTACK_COLOR, 29
POWERLINE_ERROR_COLOR, 25
POWERLINE_HARD_DIVIDER, 25
POWERLINE_HOST_COLOR, 25
POWERLINE_JOBS_COLOR, 25
POWERLINE_LOAD_COLOR, 29
POWERLINE_NEUTRAL_COLOR, 29
POWERLINE_PATH_COLOR, 26
POWERLINE_PATH_LAST_COLOR, 26
POWERLINE_PATH_SEPARATOR_COLOR, 26
POWERLINE_PROXY_COLOR, 29
POWERLINE_PROXY_MARKER, 28
POWERLINE_PYTHON_ENV_COLOR, 26
POWERLINE_PYTHON_ENV_MARKER, 25
POWERLINE_ROOT_MARKER, 25
POWERLINE_RUNTIME_COLOR, 29
POWERLINE_SECURE_MARKER, 25
POWERLINE_SOFT_DIVIDER, 25
POWERLINE_SOFTWARE_COLLECTION_MARKER, 28
POWERLINE_SOFTWARE_COLLECTIONS_COLOR, 29
POWERLINE_SPACER, 25
POWERLINE_STASH_MARKER, 25
POWERLINE_TEMPERATURE_COLOR, 29
POWERLINE_TIME_COLOR, 29
POWERLINE_USER_COLOR, 26
POWERLINE_VCS_CLEAN_COLOR, 26
POWERLINE_VCS_DIRTY_COLOR, 26
POWERLINE_VCS_MARKER, 25
POWERLINE_VCS_STASH_COLOR, 26
PROMPT_COMMAND, 6, 34
PROMPT_DIRTRIM, 53, 54
prompt_OFF()
    built-in function, 34
prompt_off()
    built-in function, 34
prompt_on()
    built-in function, 34
prompt_tag()
    built-in function, 34
PS1, 20, 30, 34, 46–49, 52, 54, 55
PWD, 53

X

XDG_CONFIG_DIRS, 7
XDG_CONFIG_HOME, 7