Liquid Prompt

Mark Vander Stel

Nov 30, 2023

CONTENTS:

Liquid Prompt 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.

If you just want a short overview of the main Liquid Prompt features, see the "*Why Liquid Prompt?*" section.

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

To know more about the available features, see the *Theming* section, which has some explanations on what's displayed on your prompt, and the *Config Options*, which lists all that change the behavior of the prompt.

For making your own theme, see the *Custom Themes* section. If you need to dig more in the code, you can browse the *Functions* section.

To check what changed in a release, check the *Release Notes*, and if you plan to do an upgrade, see *Upgrading Liquid Prompt*, and look for *deprecated options*.

CHAPTER

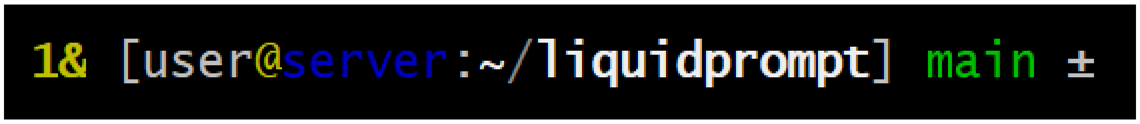
ONE

WHY LIQUID PROMPT?

Liquid Prompt gives you a **carefully designed prompt** with useful information. It shows you what you need when you need it. You will notice what changes *when* it changes, saving time and frustration. You can even use it with your favorite shell – Bash or zsh.

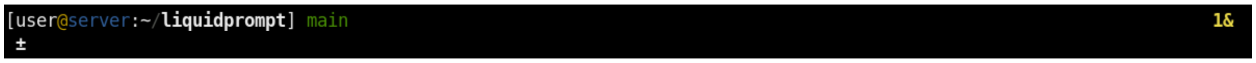
Below are screenshots of typical states that you would see in everyday use.

Using the default theme:



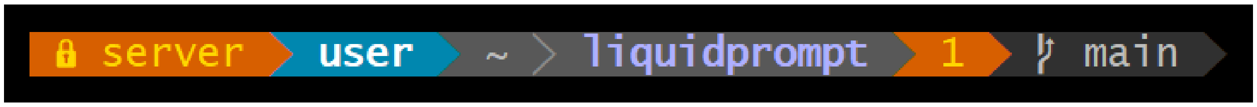
```
1& [user@server:~/liquidprompt] main ±
```

The Unfold theme, very similar to the default theme, but spread on two lines and a right-aligned section:



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

Using the Powerline theme:



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

There are many prompts configurations out there, but here is what makes Liquid Prompt stand out:

- **UX Design**: Liquid Prompt was very carefully design from the beginning, to allow for the best user experience. That is, it displays *meaningful* information with *minimal visual clutter* and *maximum readability*. While most of the other prompts are focused on aligning as much colored "segments" as possible on top of each others to look fancy, Liquid Prompt focus on what really matters.
- **Ease of use**: Liquid Prompt is written in (*heavily optimized*) portable Shell, and its installation is as easy as copying files. It also provides pre-defined themes, enabled by a simple command in your shell configuration. Configuration is as simple as editing text files with key/value pairs. No complex format to learn.
- **Shell features first**: Liquid Prompt helps you being aware of all the powerful features a modern shell can provide. While most other prompts focus on covering a lot of (boring) software version tags, Liquid Prompt shows you the features that are actually useful while you actually use the shell.
- **Configurability**: All aspects of Liquid Prompt are configurable, down to the core.

CHAPTER

TWO

FEATURES OVERVIEW

Liquid Prompt has good support of shell-related features, various version control systems, and several software configuration environments.

To browse the complete list of capabilities, see *Config Options*.

2.1 Shell Essentials

These are some of the most popular features:

- **Current path**: displays where you are, highlighting the important parts (current directory, version control repository root), smart path shortening.
- **Last command execution time**: if the last executed command took too long, display how much time it took.
- **Battery level**: show what charge remains in the battery if it's close enough to empty.
- **Username**: display the user name if it's not the login one.
- **Hostname**: indicate the hostname if over a remote connection, with host-specific color.
- **Exit code**: display the last command's exit code if it was an error.
- **Jobs**: show counters for background, sleeping, and detached jobs.
- **Time**: display the current time, using clock icons if you want a compact view.
- **Memory**: display the available memory, if not enough RAM remains.
- **Filesystem permission**: indicate if the current directory is writable or not.
- **System load**: warn if the load is too high.
- **Remote shell**: discreetly denote if you are connected through SSH, under Tmux and with X Forwarding capability.
- **Disk**: display the free disk space if it is too small.
- **Shell level**: displays the number of nested shells if it starts being too much.
- **Sudo**: show if the user currently has *sudo* rights.
- **Multiplexers**: indicate if you are in a terminal multiplexer session (i.e. tmux or screen).
- **Proxy**: indicate whether a proxy is configured.
- **Temperature**: warn if the temperature goes too high.
- **Hot prompt switch**: commands allowing you to rapidly switch the theme, go back to the default prompt, or even the most simple prompt possible.

- **Wifi**: shows the wireless network signal strength.
- **Dir stack**: helps you follow the state of this overlooked, but neat, feature of modern shells.
- **Title**: synchronize the prompt and the terminal's window title (or status bar).

2.2 Version Control

Liquid Prompt has one of the most comprehensive supports for source code management systems living in the shell. It has near-to-complete support for:

- **Git**,
- **Mercurial**,
- **Subversion**,
- **Fossil**,
- **Bazaar**.

It shows the current branch/tags, its state, and several statistics on the current commits/edits.

2.3 Software Configuration Environments

Modern development environments love to use abstractions on top of software packages. Liquid Prompt helps you knowing which one is currently in use where you are:

- **AWS**,
- **Conda**,
- **Python venv**,
- **Kubernetes**,
- **Terraform**,
- **Docker**,
- **Container**,
- **Node env**,
- **Ruby venv**,
- **Software Collections**,
- **CMake**.

Those show up only if you enter a configured directory, or have configured environment variables.

2.4 Features Disabled by Default

Some features are disabled by default, generally because they are expected to be rarely used or to not behave consistently on *all* systems.

You may want to enable those features, by setting the following configuration variables to **1** in your configuration file(s) (see the *Config Options* section to find out how to do it).

Shell essentials:

- *LP_ENABLE_DIRSTACK*
- *LP_HOSTNAME_ALWAYS*
- *LP_ENABLE_RUNTIME_BELL*
- *LP_ENABLE_SSH_COLORS*
- *LP_ENABLE_SUDO* (double-check with your sysadmin if you can enable that)
- *LP_ENABLE_DISK*
- *LP_ENABLE_ERROR_MEANING*
- *LP_ENABLE_ERROR_MEANING_EXTENDED*

Operating System:

- *LP_ENABLE_OS*
- *LP_ENABLE_OS_ARCH*
- *LP_ENABLE_OS_FAMILY*
- *LP_ENABLE_OS_DISTRIB*

Development/environments:

- *LP_ENABLE_VCS_REMOTE*
- *LP_ENV_VARS* is empty by default (but *LP_ENABLE_ENV_VARS* is enabled).
- *LP_ENABLE_CMAKE*
- *LP_ENABLE_CONTAINER* (may behave inconsistently)
- *LP_ENABLE_KUBECONTEXT*
- *LP_ENABLE_KUBE_NAMESPACE*
- *LP_ENABLE_NODE_VENV*
- *LP_ENABLE_TERRAFORM*
- *LP_ENABLE_MODULES_HASHCOLOR*

Miscellaneous:

- *LP_ENABLE_TIME*
- *LP_TIME_ANALOG*
- *LP_ENABLE_TITLE* (may behave inconsistently on exotic terminals)
- *LP_ENABLE_SCREEN_TITLE*
- *LP_ENABLE_WIFI_STRENGTH* (Linux or MacOS)
- *LP_ENABLE_HYPERLINKS* (not supported by all terminal emulators)

Disabled by default for security:

- *LP_ENABLE_VCS_ROOT* (enable at your own risk!)

2.5 Known Limitations and Bugs

- Does not display the number of commits to be pushed in Mercurial repositories.
- Browsing very large Subversion repositories may dramatically slow down the display of Liquid Prompt (use *LP_DISABLED_VCS_PATHS* to avoid that).
- Subversion repositories cannot display commits to be pushed because that's not how Subversion works.
- The window's title escape sequence may not work properly on some terminals (like *xterm-256*).
- The analog clock requires a Unicode-aware terminal and at least a sufficiently complete font on your system. The Symbola font, designed by Georges Douros, is known to work well. On Debian or Ubuntu install try the *fonts-symbola* or *ttf-ancient-fonts* package. "Powerline" and "Nerd" fonts also have the appropriate symbols.
- The `sudo` feature is disabled by default as there is no way to detect if the user has `sudo` rights without triggering a security alert that will annoy the sysadmin.

2.6 Competitors

All prompt systems tend to focus on some feature sets. If you don't like Liquid Prompt's design, you may be interested in one of those popular prompts:

- Starship: focused on showing development contexts, across various shells. Probably the largest set of detected languages.
- Spaceship: similar spirit to Starship (with a few fewer features), but only for Zsh.
- Oh-My-Posh: tries to support a little bit of everything, across various shells, largest set of themes (albeit not very diverse).
- Powerlevel10k: the most popular on Github, focused on fancy features (like transitive prompt), but to the expense of the UX. Only supports Zsh.
- Powerline: primarily a daemon-based status line generator, hence with good support for latency-expensive features. Probably the largest feature set for various services status.
- Pure: quite popular, despite focusing on being minimalist (only supports Zsh, Git, and a few shell features).

The following table compares those prompts systems in details.

Warning: This information has been gathered by *nojhan*, one of the authors of Liquid Prompt. As such, it is highly subjective. Judgments made about the levels of support are extremely arbitrary. Take this with a grain of salt.

In this table, the numbers in cells figure the level of quality of the feature. *Popularity* is the sum of levels in the row. *Support* lines are the sum of levels in the column, for each *category* section. *Category* sections are sorted from top to bottom based on their average popularity. Projects are sorted from left to right, based on their *support* score in the *essentials* section.

2.7 License

Liquid Prompt is distributed under the GNU Affero General Public License version 3.

To comply with the AGPL clauses, anybody offering Liquid Prompt over the network is *required* to also offer access to the source code of it and allow further use and modifications. As Liquid Prompt is implemented purely in shell script, anybody using it over SSH or equivalent terminal connection automatically also has access to the source code, **so it is easy to comply with the license**.

The only case in which you may violate the license is if you provide a shell service but do not allow the user to download your Liquid Prompt version. For instance if you offer an access to a virtual machine through a graphical session, without allowing files transfer. In that case, you are required to explicitly indicate to your users where they may download the code that is running your version of Liquid Prompt (even if you only provide a theme on top of the base code).

CHAPTER

THREE

INSTALLATION

3.1 Packages

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

3.1.1 Latest Versions

Source: repology.org.

3.1.2 Install commands

Archlinux

```
pacman -S liquidprompt
```

Debian

… and Debian derivatives.

```
apt-get install liquidprompt
```

A small script, `liquidprompt_activate` (not to be confused with *`lp_activate()`*) is included to ease activation of the prompt, which can be used instead of the *Shell Installation* instructions.

This will set the required environment:

- The files `~/.bashrc` and/or `~/.zshrc` are modified to load Liquid Prompt at startup.
- If no previous `~/.config/liquidpromptrc` file exists, it will be created.

So, to get Liquid Prompt working simply run:

```
liquidprompt_activate
source ~/.bashrc  # or ~/.zshrc
```

Be aware that multiple invocations of the `liquidprompt_activate` command may pollute `~/.bashrc` and/or `~/.zshrc` files.

Homebrew

```
brew install liquidprompt
```

Nix

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

3.2 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 --branch stable https://github.com/liquidprompt/liquidprompt.git ~/liquidprompt
```

Or, if you want to use the development (non-stable) branch:

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

3.2.1 Installation via Antigen

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

```
antigen bundle liquidprompt/liquidprompt
```

3.2.2 Installation via Zinit

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

```
zinit ice ver"stable" lucid nocd
zinit light liquidprompt/liquidprompt
```

3.3 Dependencies

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

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

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.

- Terminal formatting requires `tput`.
- Time display requires `date`.
- Detached session status looks for `screen` and/or `tmux`.
- VCS support features require `git`, `hg`, `svn`, `bzr` or `fossil` for their respective repositories.

3.4 Test Drive

To test the prompt immediately after download, run:

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

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

3.5 Shell Installation

To use Liquid Prompt 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 Liquid Prompt in interactive shells, not from a script or from scp
[[ $- = *i* ]] && source ~/liquidprompt/liquidprompt
```

Adjust the path if you installed to a different location than 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 Liquid Prompt to avoid history and timing issues. Do not export `PROMPT_COMMAND`.

Warning: If you are using bash-preexec, be aware that bash-preexec **must** come **before** liquidprompt in your `.bashrc`. This contradicts their documentation, which says "[bash-preexec] must be the last thing imported in your bash profile", but since Liquid Prompt special-cases bash-preexec, it must be loaded after bash-preexec.

Next up are the *Config Options*.

CHAPTER

FOUR

CONFIG OPTIONS

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

To get your own configuration, you may want to generate a default configuration by calling the following script:

```
./tools/config-from-doc.sh > my_liquidpromptrc
```

Then edit the `my_liquidpromptrc` file to suits your needs and copy/link it where you want.

In the event that you synchronize your configuration file across multiple computers, or if you have an `/etc/liquidpromptrc` system-wide from which you'd like to make minor deviations in an individual user account, you can augment the primary config to add in any local modifications using lines such as these:

```
LOCAL_RCFILE=$HOME/.liquidpromptrc.local
[ -f "$LOCAL_RCFILE" ] && source "$LOCAL_RCFILE"
```

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.

Several example of configurations are given in the `contrib/presets` directory. Some of these presets can be combined, for instance for changing the icons, along with the colors.

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

4.1 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*, or add a newline before the prompt mark.

For example:

```
LP_MARK_PREFIX=$'\n'
```

`LP_PATH_CHARACTER_KEEP: int = 3`

The number of characters to save at the start and possibly the end of a directory name when shortening the path. See *LP_PATH_METHOD* for details of the specific methods.

New in version 2.0.

`LP_PATH_DEFAULT: string = ""`

Deprecated since version 2.0: Use *LP_PATH_METHOD* set to *truncate_to_last_dir* instead.

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 (counting '/') 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.

LP_ENABLE_SHORTEN_PATH must be enabled to have any effect.

See also: *LP_PATH_LENGTH* and *LP_PATH_METHOD*.

Changed in version 2.0: No longer supports a value of `-1`.

`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.

Note: *LP_PATH_KEEP* and *LP_PATH_METHOD* have higher precedence over this option. Important path parts, including directories saved by *LP_PATH_KEEP*, *LP_PATH_VCS_ROOT*, and the last directory, will always be displayed, even if the path does not fit in the maximum length.

LP_PATH_METHOD: string = "truncate_chars_from_path_left"

Sets the method used for shortening the path display when it exceeds the maximum length set by *LP_PATH_LENGTH*.

- **truncate_chars_from_path_left**: Truncates characters from the start of the path, showing consecutive directories as one shortened section. E.g. in a directory named `~/MyProjects/liquidprompt/tests`, it will be shortened to `...prompt/tests`. The shortened mark is *LP_MARK_SHORTEN_PATH*.
- **truncate_chars_from_dir_right**: Leaves the beginning of a directory name untouched. E.g. directories will be shortened like so: `~/Doc.../Office`. How many characters will be untouched is set by *LP_PATH_CHARACTER_KEEP*. The shortened mark is *LP_MARK_SHORTEN_PATH*.
- **truncate_chars_from_dir_middle**: Leaves the beginning and end of a directory name untouched. E.g. in a directory named `~/MyProjects/Office`, then it will be shortened to `~/MyP...cts/Office`. How many characters will be untouched is set by *LP_PATH_CHARACTER_KEEP*. The shortened mark is *LP_MARK_SHORTEN_PATH*.
- **truncate_chars_to_unique_dir**: Truncate each directory to the shortest unique starting portion of their name. E.g. in a folder `~/dev/liquidprompt`, it will be shortened to `~/d/liquidprompt` if there is no other directory starting with 'd' in the home directory.
- **truncate_to_last_dir**: Only display the last directory in the path. In other words, the current directory name.

All methods (other than *truncate_to_last_dir*) start at the far left of the path (limited by *LP_PATH_KEEP*). Only the minimum number of directories needed to fit inside *LP_PATH_LENGTH* will be shortened.

LP_ENABLE_SHORTEN_PATH must be enabled to have any effect.

New in version 2.0.

LP_PATH_VCS_ROOT: bool = 1

Display the root directory of the current VCS repository with special formatting, set by *LP_COLOR_PATH_VCS_ROOT*. If *LP_ENABLE_SHORTEN_PATH* is enabled, also prevent the path shortening from shortening or hiding the VCS root directory.

New in version 2.0.

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()*.

LP_TIME_FORMAT: string = "%H:%M:%S"

The formatting string passed to *date(1)* using formatting from *strftime(3)* used to display the current date and/or time.

See also: *LP_ENABLE_TIME*.

New in version 2.1.

4.2 Features

`LP_ALWAYS_DISPLAY_VALUES: bool = 1`

Display the actual values of load, batteries, and wifi signal strength along with their corresponding marks. Disable to only print the colored marks.

See also: *`LP_ENABLE_LOAD`*, *`LP_ENABLE_BATT`*, *`LP_ENABLE_DISK`*, and *`LP_ENABLE_WIFI_STRENGTH`*.

`LP_DELIMITER_KUBECONTEXT_PREFIX: string = ""`

Delimiter to shorten the Kubernetes context by removing a prefix.

Usage example:

- if your context names are *cluster-dev* and *cluster-test*, then set this to "-" in order to output *dev* and *test* in prompt.
- if using AWS EKS then set this to "/" to show only the cluster name, without the rest of the ARN (`arn:aws:eks:$AWS_REGION:$ACCOUNT_ID:cluster/$CLUSTER_NAME`)
- alternatively, if using AWS EKS, set this to ":" to show only *cluster/$CLUSTER_NAME*. (Note: the prefix removed is a greedy match - it contains all the ":"s in the input.)

If set to the empty string no truncating will occur (this is the default).

See also: *`LP_ENABLE_KUBECONTEXT`*, *`LP_DELIMITER_KUBECONTEXT_SUFFIX`*, *`LP_COLOR_KUBECONTEXT`*, and *`LP_MARK_KUBECONTEXT`*.

New in version 2.1.

`LP_DELIMITER_KUBECONTEXT_SUFFIX: string = ""`

Delimiter to shorten the Kubernetes context by removing a suffix.

Usage example:

- if your context names are *dev-cluster* and *test-cluster*, then set this to "-" in order to output *dev* and *test* in prompt.
- if your context names are *dev.k8s.example.com* and *test.k8s.example.com*, then set this to "." in order to output *dev* and *test* in prompt. (Note: the suffix removed is a greedy match - it contains all the "."s in the input.)
- if using OpenShift then set this to "/" to show only the project name without the cluster and user parts.

If set to the empty string no truncating will occur (this is the default).

See also: *`LP_ENABLE_KUBECONTEXT`*, *`LP_DELIMITER_KUBECONTEXT_PREFIX`*, *`LP_COLOR_KUBECONTEXT`*, and *`LP_MARK_KUBECONTEXT`*.

New in version 2.1.

`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*.

New in version 2.0.

LP_DISPLAY_VALUES_AS_PERCENTS: bool = 0

When displaying a value, show it as a percentage if possible.

Used in sensors for capacities, see *LP_ENABLE_DISK*, *LP_ENABLE_BATT*.

LP_ENABLE_AWS_PROFILE: bool = 1

Display the current value of `AWS_PROFILE`, `AWS_DEFAULT_PROFILE`, or `AWS_VAULT`. AWS_PROFILE and AWS_DEFAULT_PROFILE are used to switch between configuration profiles by the AWS CLI. AWS_VAULT is used by aws-vault to specify the AWS profile in use.

See also: *LP_COLOR_AWS_PROFILE*.

New in version 2.1.

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_ALWAYS_DISPLAY_VALUES* is enabled.

Will be disabled if `acpi` is not found on Linux, fails to read the Linux sysfs system, 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_CHROOT: bool = 1

Display whether a *chroot* environment is active.

LP_ENABLE_CMAKE: bool = 0

Displays the current configuration of CMake, if the directory contains a *CMakecache.txt*. Displays the compiler, the generator and the build type, separated by *LP_MARK_CMAKE*.

Will be disabled if `cmake` is not found.

The compiler is displayed without its path. The generator is displayed without space, and some names are shortened (*Makefiles* as *Make* and *Visual Studio* as *VS*), so that, for instance: *Unix Makefiles* will be displayed as *UnixMake*. Both fields are randomly colored according to their hash.

The common build type colors can be configured:

- *Debug*, colored with *LP_COLOR_CMAKE_DEBUG* (magenta, by default),
- *RelWithDebInfo*, colored with *LP_COLOR_CMAKE_RWDI* (blue, by default),
- *Release*, colored with *LP_COLOR_CMAKE_RELEASE* (cyan, by default),
- any other value would be colored according to its hash.

New in version 2.2.

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_CONTAINER: bool = 0

Indicate if the shell is running in a container environment (e.g. Docker, Podman, LXC, Singularity, systemd-nspawn).

Note: Containers may inherit some or even no variables from their parent shell, so this may behave inconsistently with different container software. For example, Docker does not inherit anything unless explicitly told to. Singularity in many configurations inherits most variables but shell functions and zsh hooks might not make it in. For full functionality, `liquidprompt` may need to be sourced inside the child container.

See also: *LP_COLOR_CONTAINER*.

New in version 2.1.

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_DISPLAY: bool = 1

Detect if the connection has X11 support.

In the default theme, display a green `@` if it does; a yellow one if not.

See also *LP_COLOR_X11_ON* and *LP_COLOR_X11_OFF*.

New in version 2.2.

LP_ENABLE_DISK: bool = 1

Display *LP_MARK_DISK* if the free space on the hard drive hosting the current directory goes below a threshold.

Thresholds can be stated either:

- as a percentage with *LP_DISK_THRESHOLD_PERC*,
- or an absolute number *of kilobytes* with *LP_DISK_THRESHOLD*.

Display will occur if one of the thresholds is met.

If *LP_ALWAYS_DISPLAY_VALUES* is enabled, the prompt will show the available space along with *LP_MARK_DISK*, if disabled, it will show only the mark.

The precision of the available space can be configured with *LP_DISK_PRECISION*.

If *LP_DISPLAY_VALUES_AS_PERCENTS* is enabled, it will show the percentage, if it is disabled, it will show the absolute value in a human-readable form (i.e. with metric prefixed units).

New in version 2.2.

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_ERROR_MEANING: bool = 0

Display a guess on the last error meaning.

Note: This only enable a limited subset of error codes, that are very probably in use on several systems. To enable more codes (and probably more false positives) see *LP_ENABLE_ERROR_MEANING_EXTENDED*.

See also: *LP_COLOR_ERR*.

New in version 2.2.

LP_ENABLE_ERROR_MEANING_EXTENDED: bool = 0

Extends the set of interpreted error codes to a larger set of (POSIX) codes.

Note: This use a reasonable set of error codes that are common on POSIX systems on x86 or ARM architectures (most notably from `sysexit.h` and `signal.h`). But any software may use its own set of codes, and thus the guess may be wrong.

This has no effect if *LP_ENABLE_ERROR_MEANING* is disabled. See also: *LP_COLOR_ERR*.

New in version 2.2.

LP_ENABLE_ENV_VARS: bool = 1

Display a user-defined set of environment variables. May show if the variables are unset, set, or their actual content.

Watched variables should be added to the *LP_ENV_VARS* array.

The resulting prompt section is configured by:

- *LP_MARK_ENV_VARS_OPEN*
- *LP_MARK_ENV_VARS_SEP*
- *LP_MARK_ENV_VARS_CLOSE*
- *LP_COLOR_ENV_VARS_SET*
- *LP_COLOR_ENV_VARS_UNSET*

New in version 2.2.

`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`

Deprecated since version 2.1: Use *`LP_HOSTNAME_METHOD`* set to "full" instead.

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

Note: This never functioned as intended, and would only show the FQDN if `/etc/hostname` contained the full domain name. For a more portable and reliable version, set *`LP_HOSTNAME_METHOD`* to *fqdn*.

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`* and *`LP_HG_COMMAND`*.

`LP_ENABLE_HYPERLINKS: bool = 0`

Adds clickable links to some elements of the prompt:

- If locally connected, adds a link to each displayed elements of the path, using the `file://` scheme.
- Within remote SSH connections, adds a link to each element of the path, but using the `sftp://` protocol, configured with the *current* username and hostname.
- If the hostname is displayed within an SSH connection, adds a `ssh://` URL to it.

The links take the form of a OSC-8 escape sequences containing an Uniform Resource Locator, which should be interpreted by the terminal emulator. If your terminal emulator does not support OSC-8, it may display escapement garbage. As not all terminal emulator support links, this feature is disabled by default.

Warning: Your system should be configured to handle the aforementioned link schemes. If nothing happen when you click on the link, or if the wrong application is used, there is a configuration problem on your system or with your terminal emulator (not with Liquid Prompt).

Note: Liquid Prompt cannot possibly follow complex remote connections. Remote links are thus configured with the *current* username, and the *current* fully qualified domain name, as `sftp://<username>@<hostname>/<path>`. It is possible that this URL does not work the same way than a manual connection. For instance, if you proxy jumped (i.e. if you jumped from one connection to the other), and/or you logged in with another user, and/or used SSH aliases, then the links probably won't work the way you may expect.

New in version 2.2.

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_KUBECONTEXT: bool = 0

Display the current Kubernetes context.

See also: *LP_ENABLE_KUBE_NAMESPACE*, *LP_DELIMITER_KUBECONTEXT_PREFIX*, *LP_DELIMITER_KUBECONTEXT_SUFFIX*, *LP_COLOR_KUBECONTEXT*, and *LP_MARK_KUBECONTEXT*.

New in version 2.1.

LP_ENABLE_KUBE_NAMESPACE: bool = 0

Display the current Kubernetes default namespace in the current context.

See also: *LP_ENABLE_KUBECONTEXT*, *LP_DELIMITER_KUBECONTEXT_PREFIX*, *LP_DELIMITER_KUBECONTEXT_SUFFIX*, *LP_COLOR_KUBECONTEXT*, and *LP_MARK_KUBECONTEXT*.

New in version 2.1.

LP_ENABLE_LOAD: bool = 1

Display the load average over the past 1 minutes when above the threshold.

See also: *LP_LOAD_THRESHOLD*, *LP_LOAD_CAP*, *LP_MARK_LOAD*, *LP_ALWAYS_DISPLAY_VALUES*, and *LP_COLORMAP*. *LP_MARK_LOAD*, *LP_PERCENTS_ALWAYS*, and *LP_COLORMAP*.

LP_ENABLE_MODULES: bool = 1

Display the currently loaded Modules.

See also: * *LP_ENABLE_MODULES_VERSIONS*, * *LP_ENABLE_MODULES_HASHCOLOR*, * *LP_COLOR_MODULES*, * *LP_MARK_MODULES_OPEN*, * *LP_MARK_MODULES_SEP*, * *LP_MARK_MODULES_CLOSE*.

New in version 2.2.

LP_ENABLE_MODULES_VERSIONS: bool = 1

Display the currently loaded modules' versions, after their names (separated by a slash, as in the `module list` command).

If disabled, only the name of the module is displayed.

See *LP_ENABLE_MODULES*.

New in version 2.2.

LP_ENABLE_MODULES_HASHCOLOR: bool = 0

If enabled, each item in the modules section will be randomly colored, according to its hash, instead of using *LP_COLOR_MODULES*.

See *LP_ENABLE_MODULES*.

New in version 2.2.

LP_ENABLE_MULTIPLEXER: bool = 1

Allows getting the name of the current multiplexer (*screen* or *tmux*), if any.

If set to `0`, also disables:

- *LP_COLOR_IN_MULTIPLEXER*,
- *LP_MARK_MULTIPLEXER_OPEN* and *LP_MARK_MULTIPLEXER_CLOSE*.

New in version 2.2.

`LP_ENABLE_NODE_VENV: bool = 0`

Display the currently activated nodeenv or NVM virtual environment.

See also: *LP_COLOR_NODE_VENV*.

New in version 2.1.

`LP_ENABLE_OS: bool = 0`

Display information about the current Operating System.

Degree of details is controlled by:

- *LP_ENABLE_OS_ARCH*
- *LP_ENABLE_OS_FAMILY*
- *LP_ENABLE_OS_KERNEL*
- *LP_ENABLE_OS_DISTRIB*
- *LP_ENABLE_OS_VERSION*

Note: As of now, only Linux may have detailed information about the distribution and version.

See also *LP_MARK_OS* and *LP_MARK_OS_SEP* for configuring the appearance.

If no replacement string is provided with *LP_MARK_OS*, each item will be randomly colored, according to its hash.

New in version 2.2.

`LP_ENABLE_OS_ARCH: bool = 0`

Display the processor architecture of the current OS.

See *LP_ENABLE_OS*.

New in version 2.2.

`LP_ENABLE_OS_DISTRIB: bool = 0`

Display the current Linux distribution.

See *LP_ENABLE_OS*.

New in version 2.2.

`LP_ENABLE_OS_FAMILY: bool = 0`

Display the family of the current OS (UNIX, BSD, GNU, or Windows).

See *LP_ENABLE_OS*.

New in version 2.2.

`LP_ENABLE_OS_KERNEL: bool = 1`

Display the name of the kernel type for the current OS.

This may be “Linux”, “FreeBSD”, “SunOS”, “Darwin”, “Cygwin”, “MSYS”, “MinGW”, “OpenBSD”, “DragonFly”.

See *LP_ENABLE_OS*.

New in version 2.2.

LP_ENABLE_OS_VERSION: bool = 1

Display the version "codename" of the current Linux distribution.

See *LP_ENABLE_OS*.

New in version 2.2.

LP_ENABLE_PATH: bool = 1

Display the current working directory.

New in version 2.2: Before this version, this feature was always enabled.

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_RAM: bool = 1

Display a *LP_MARK_RAM* mark when the available amount of Random Access Memory goes below a threshold.

Thresholds can be stated either:

- as a percentage with *LP_RAM_THRESHOLD_PERC*,
- or an absolute number *of kilobytes* with *LP_RAM_THRESHOLD*.

Display will occur if one of the thresholds is met.

If *LP_ALWAYS_DISPLAY_VALUES* is enabled, the prompt will show the available space along with *LP_MARK_RAM*, if disabled, it will show only the mark.

The precision of the displayed available space can be configured with *LP_RAM_PRECISION*.

If *LP_DISPLAY_VALUES_AS_PERCENTS* is enabled, it will show the percentage, if it is disabled, it will show the absolute value in a human-readable form (i.e. with metric prefixed units).

New in version 2.2.

LP_ENABLE_RUBY_VENV: bool = 1

Display the currently activated RVM or RBENV virtual environment.

See also: *LP_RUBY_RVM_PROMPT_OPTIONS* and *LP_COLOR_RUBY_VENV*.

New in version 2.1.

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_SHLVL: bool = 1

Show the value of `$SHLVL`, which is the number of nested shells. For example, if one runs `bash` inside their shell, it will open a new shell inside their current shell, and this will display "2".

See also: *LP_MARK_SHLVL* and *LP_COLOR_SHLVL*.

New in version 2.1.

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_METHOD*, *LP_PATH_LENGTH*, *LP_PATH_KEEP*, *LP_PATH_CHARACTER_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, unless you have *NOPASSWD* powers. 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, or fails to read from the Linux sysfs system.

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

LP_ENABLE_TERRAFORM: bool = 0

Display the currently activated Terraform workspace.

See also: *LP_COLOR_TERRAFORM*.

New in version 2.1.

LP_ENABLE_TIME: bool = 0

Displays the time at which the prompt was shown. The format can be configured with *LP_TIME_FORMAT*.

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_TITLE_COMMAND: bool = 1

Postpend the currently running command to the terminal title while the command is running.

LP_ENABLE_TITLE must be enabled to have any effect.

New in version 2.1.

LP_ENABLE_TMUX_TITLE_PANES: bool = 1

Sets the title of the Tmux pane instead of the window.

LP_ENABLE_TITLE and *LP_ENABLE_SCREEN_TITLE* must be enabled to have any effect.

New in version 2.2.

LP_ENABLE_VCS_REMOTE: bool = 0

Enable the display of the remote repository in the VCS state section.

If enabled, will display *LP_MARK_VCS_REMOTE*, followed by the remote repository name.

In the default theme, if the remote repository has commits not pulled in the local branch, the mark will be showed in *LP_COLOR_COMMITS_BEHIND*. If the local repository has commits not pushed to the remote branch, the remote name is shown in *LP_COLOR_COMMITS*. If neither is the case, nothing will be shown.

New in version 2.2.

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_ENABLE_WIFI_STRENGTH: bool = 0

Display an indicator if any wireless signal strength percentage is below *LP_WIFI_STRENGTH_THRESHOLD*. Also show the strength percentage if *LP_ALWAYS_DISPLAY_VALUES* is enabled.

Both Linux and MacOS are supported.

See also: *LP_MARK_WIFI* and *LP_COLORMAP*.

New in version 2.1.

LP_ENV_VARS: array<string> = ()

The set of environment variables that the user wants to watch.

Items should be a string with three space-separated elements of the form *"<name> <set>[<unset>]"*, containing:

- the name of the variable to watch,
- the string to display if the variable is set,
- (optionally) the string to display if the variable is not set.

The string used when the variable is set may contain the `%s` mark, which is replaced by the actual content of the variable.

For example:

```
LP_ENV_VARS=(
    # Display "V" if VERBOSE is set, nothing if it's unset.
    "VERBOSE V"
    # Display the name of the desktop session, if set, T if unset.
    "DESKTOP_SESSION %s T"
    # Display "ed:" followed the name of the default editor, nothing if unset.
    "EDITOR ed:%s"
)
```

See also *LP_ENABLE_ENV_VARS*.

The resulting prompt section is configured by:

- *LP_MARK_ENV_VARS_OPEN*
- *LP_MARK_ENV_VARS_SEP*
- *LP_MARK_ENV_VARS_CLOSE*
- *LP_COLOR_ENV_VARS_SET*
- *LP_COLOR_ENV_VARS_UNSET*

`LP_HG_COMMAND: string = "hg"`

The command to use for Mercurial commands. Can be used to replace `hg` with `rhg` or `chg`.

See also: *LP_ENABLE_HG* and *LP_MARK_HG*.

New in version 2.1.

`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_HOSTNAME_METHOD: string = "short"`

Determine the method for displaying the hostname.

- **short**: show the first section of the hostname, what is before the first dot. Equal to `\h` in Bash or `%m` in Zsh.
- **full**: show the full hostname, without any domain name. Equal to `\H` in Bash or `%M` in Zsh.
- **fqdn**: show the fully qualified domain name, if it exists. Defaults to **full** if not.
- **pretty**: show the pretty hostname, also called "machine display name". Defaults to **full** if one does not exist.

See also: *LP_HOSTNAME_ALWAYS*.

New in version 2.1.

LP_PERCENTS_ALWAYS: bool = 1

Deprecated since version 2.2: Use *LP_ALWAYS_DISPLAY_VALUES* and *LP_DISPLAY_VALUES_AS_PERCENTS* instead.

Display the actual values of load, batteries, and wifi signal strength along with their corresponding marks. Disable to only print the colored marks.

See also: *LP_ENABLE_LOAD*, *LP_ENABLE_BATT*, *LP_ENABLE_WIFI_STRENGTH*.

LP_RUBY_RVM_PROMPT_OPTIONS: array<string> = (i v g s)

An array of single letter switches to customize the RVM prompt output.

Will only have an effect if *LP_ENABLE_RUBY_VENV* is enabled and you are using RVM (i.e. no effect with RBENV).

New in version 2.1.

LP_TEMP_SYSFS_IGNORE_FILES: array<string> = ()

Paths to files in the sysfs interface that should be ignored when reading temperature sensors. A path can include globs.

See also *LP_ENABLE_TEMP*.

New in version 2.2.

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.

4.3 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_DISK_PRECISION: int = 2`

Control the numbers of decimals when displaying the absolute available space of the current hard drive. If set to 0, don't display decimals. If set to 1 or 2, display decimals.

See *LP_ENABLE_DISK*, *LP_ALWAYS_DISPLAY_VALUES*, and *LP_DISPLAY_VALUES_AS_PERCENTS*.

New in version 2.2.

`LP_DISK_THRESHOLD: int = 100000`

Display something if the available space on the hard drive hosting the current directory goes below this absolute threshold *in kilobytes*.

The threshold for disk can also be set with *LP_DISK_THRESHOLD_PERC*, the first one to be reached triggering the display.

See also *LP_ENABLE_DISK*.

New in version 2.2.

`LP_DISK_THRESHOLD_PERC: int = 10`

Display something if the available space on the hard drive hosting the current directory goes below this percentage.

The threshold for disk can also be set with *LP_DISK_THRESHOLD*, the first one to be reached triggering the display..

See also *LP_ENABLE_DISK*.

New in version 2.2.

`LP_LOAD_CAP: float = 2.0`

The value for load average per CPU to display with the max color scaling. Values above this number will still be displayed, but the colors will not increase in intensity.

LP_ENABLE_LOAD must be enabled to have any effect.

See also: *LP_COLORMAP*.

New in version 2.0.

`LP_LOAD_THRESHOLD: float = 0.60`

Display the load average per CPU when above this threshold. For historical reasons, this number must have a decimal point ('.'), or it will be treated as a percentage.

LP_ENABLE_LOAD must be enabled to have any effect.

Changed in version 2.0: Accepts float values of actual load averages. Integer values of centiload are still accepted, but deprecated.

`LP_RAM_PRECISION: int = 2`

Control the numbers of decimals when displaying the absolute available space of the current system RAM. If set to 0, don't display decimals. If set to 1 or 2, display decimals.

See *LP_ENABLE_RAM*, *LP_ALWAYS_DISPLAY_VALUES*, and *LP_DISPLAY_VALUES_AS_PERCENTS*.

New in version 2.2.

`LP_RAM_THRESHOLD: int = 100000`

Display something if the available RAM space goes below this absolute threshold *in kilobytes*.

The threshold for RAM can also be set with *LP_RAM_THRESHOLD_PERC*, the first one to be reached triggering the display.

See also *LP_ENABLE_RAM*.

New in version 2.2.

`LP_RAM_THRESHOLD_PERC: int = 10`

Display something if the available RAM space goes below this percentage.

The threshold for RAM can also be set with *`LP_RAM_THRESHOLD`*, the first one to be reached triggering the display..

See also *`LP_ENABLE_RAM`*.

New in version 2.2.

`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.

`LP_WIFI_STRENGTH_THRESHOLD: int = 40`

Display the lowest wireless signal strength when the strength percentage is below this threshold.

`LP_ENABLE_WIFI_STRENGTH` must be enabled to have any effect.

New in version 2.1.

4.4 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_MULTIPLEXER_CLOSE`*.

Changed in version 2.2: Can be disabled by *`LP_ENABLE_MULTIPLEXER`*.

`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_MULTIPLEXER_OPEN*.

Changed in version 2.2: Can be disabled by *LP_ENABLE_MULTIPLEXER*.

`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_CMAKE: string = ":"`

Separator used for fields of *LP_ENABLE_CMAKE*.

New in version 2.2.

`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_DEV_CLOSE: string = ">"`

Closing of the “development tools” section.

New in version 2.2.

`LP_MARK_DEV_MID: string = "|"`

Separator between elements of the “development tools” section.

New in version 2.2.

`LP_MARK_DEV_OPEN: string = "<"`

Opening of the “development tools” section.

New in version 2.2.

`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_DISK: string = " "`

Mark used to indicate that the available disk space is too low. See *LP_ENABLE_DISK*.

New in version 2.2.

`LP_MARK_ENV_VARS_OPEN: string = "("`

Mark used to start the user-defined environment variables watch list.

See also:

- *LP_ENABLE_ENV_VARS*
- *LP_ENV_VARS*

- *LP_MARK_ENV_VARS_SEP*
- *LP_MARK_ENV_VARS_CLOSE*
- *LP_COLOR_ENV_VARS_SET*
- *LP_COLOR_ENV_VARS_UNSET*

New in version 2.2.

LP_MARK_ENV_VARS_SEP: string = " "

Mark used to separate items of the user-defined environment variables watch list.

See also:

- *LP_ENABLE_ENV_VARS*
- *LP_ENV_VARS*
- *LP_MARK_ENV_VARS_OPEN*
- *LP_MARK_ENV_VARS_CLOSE*
- *LP_COLOR_ENV_VARS_SET*
- *LP_COLOR_ENV_VARS_UNSET*

New in version 2.2.

LP_MARK_ENV_VARS_CLOSE: string = ")"

Mark used to end the user-defined environment variables watch list.

See also:

- *LP_ENABLE_ENV_VARS*
- *LP_ENV_VARS*
- *LP_MARK_ENV_VARS_OPEN*
- *LP_MARK_ENV_VARS_SEP*
- *LP_COLOR_ENV_VARS_SET*
- *LP_COLOR_ENV_VARS_UNSET*

New in version 2.2.

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* and *LP_HG_COMMAND*.

LP_MARK_JOBS_SEPARATOR: string = "/"

Mark used to separate elements of *LP_JOBS*.

See also *LP_ENABLE_JOBS*.

New in version 2.2.

LP_MARK_KUBECONTEXT: string = ""

Mark used to prefix the current Kubernetes context.

Used to visually distinguish the Kubernetes context from other context fields like the Python virtual environment (see *LP_ENABLE_VIRTUALENV*) and the Red Hat Software Collection (see *LP_ENABLE_SCLS*).

The display of Unicode characters varies among Terminal and Font settings, so you might try alternative marks. Single symbol alternatives to the default (U+2388, Helm Symbol) are (U+2638, Wheel of Dharma) or (U+03BA, Greek Small Letter Kappa).

See also: *LP_ENABLE_KUBECONTEXT*.

New in version 2.1.

LP_MARK_LOAD: string = ""

Mark used before displaying load average.

See also: *LP_ENABLE_LOAD*.

LP_MARK_MODULES_OPEN: string = ""

Mark used before displaying loaded modules.

See also: *LP_ENABLE_MODULES*.

LP_MARK_MODULES_CLOSE: string = ""

Mark used after displaying loaded modules.

See also: *LP_ENABLE_MODULES*.

LP_MARK_MODULES_SEP: string = ":"

Mark used between loaded modules.

See also: *LP_ENABLE_MODULES*.

LP_MARK_MULTIPLEXER_CLOSE: string = $LP_MARK_BRACKET_CLOSE

Mark used for closing core prompt brackets. Used by the default theme when inside of a multiplexer.

See also: *LP_MARK_MULTIPLEXER_OPEN*, *LP_MARK_BRACKET_CLOSE*.

New in version 2.1.

Changed in version 2.2: Can be disabled by *LP_ENABLE_MULTIPLEXER*.

LP_MARK_MULTIPLEXER_OPEN: string = $LP_MARK_BRACKET_OPEN

Mark used for opening core prompt brackets. Used by the default theme when inside of a multiplexer.

See also: *LP_MARK_MULTIPLEXER_CLOSE*, *LP_MARK_BRACKET_OPEN*.

New in version 2.1.

Changed in version 2.2: Can be disabled by *LP_ENABLE_MULTIPLEXER*.

LP_MARK_OS: array<string> = ()

A list of pair of strings to be replaced by another string when displaying information about the OS.

Each pair in the list configures the match, then the replacement string.

For instance, if you set `LP_MARK_OS=("Linux" "L")` and `LP_ENABLE_OS=1 ; LP_ENABLE_OS_FAMILY=1`, then any occurrence of "Linux" will be replaced by an "L" in the OS section.

It is possible to use presets colors in the replacement string (see the *Colors* section below). Note that if a replacement occurs, the result will *not* be colored automatically.

For example, to shorten known names, you can use the following configuration (if your font supports those characters):

```
LP_MARK_OS=(
    # Arch
    "x86_64"    "${BLUE}x64${NO_COL}"
    "i386"      "i3"
    "i686"      "i6"
    "aarch64"   "${GREEN}a64${NO_COL}"
    # Families
    "BSD"       "${RED}BSD${NO_COL}"
    "Windows"   ""
    "Unix"      "U"
    "GNU"       ""
    # Kernels
    "FreeBSD"   ""
    "DragonFly" ""
    "OpenBSD"   ""
    "Darwin"    ""
    "SunOS"     "${BOLD_YELLOW}${NO_COL}"
    "Cygwin"    ""
    "MSYS"      "M"
    "MinGW"     "GW"
    "Linux"     ""
)
```

See *LP_ENABLE_OS*.

New in version 2.2.

LP_MARK_OS_SEP: string = "/"

The character used to separate items of the OS section.

See *LP_ENABLE_OS*.

New in version 2.2.

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_RAM: string = M

Mark used before displaying available Random Access Memory. See *LP_ENABLE_RAM*.

New in version 2.2.

`LP_MARK_SHLVL: string = "└"`

Mark used to indicate the shell is inside another shell.

See also: *`LP_ENABLE_SHLVL`* and *`LP_COLOR_SHLVL`*.

New in version 2.1.

`LP_MARK_SHORTEN_PATH: string = " ... "`

Mark used to indicate a portion of the path was hidden to save space. Not all shortening methods use this mark, some only use *`LP_COLOR_PATH_SHORTENED`*.

See also: *`LP_ENABLE_SHORTEN_PATH`*, *`LP_PATH_METHOD`*.

`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_VCS_REMOTE: string = ""`

Mark used to indicate the VCS remote repository name and status.

See *`LP_ENABLE_VCS_REMOTE`*.

New in version 2.2.

`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.

`LP_MARK_WIFI: string = ""`

Mark used before displaying wireless signal strength.

See also: *`LP_ENABLE_WIFI_STRENGTH`*.

New in version 2.1.

4.5 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> = ("" $GREEN $BOLD_GREEN $YELLOW $BOLD_YELLOW $RED $BOLD_RED $WARN_RED $CRIT_RED $DANGER_RED)

An array of colors that is used by the battery, load, temperature, and wireless signal strength 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.

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

LP_COLOR_AWS_PROFILE: string = $YELLOW

Color used to display the current active AWS Profile.

See also: *LP_ENABLE_AWS_PROFILE*.

New in version 2.1.

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_CMAKE_BUILD: string = $MAGENTA

Color used to display the build type in the CMake segment.

See *LP_ENABLE_CMAKE*.

New in version 2.2.

LP_COLOR_CMAKE_C: string = $MAGENTA

Color used to display the C compiler in the CMake segment.

See *LP_ENABLE_CMAKE*.

New in version 2.2.

LP_COLOR_CMAKE_CXX: string = $MAGENTA

Color used to display the C++ compiler in the CMake segment.

See *LP_ENABLE_CMAKE*.

New in version 2.2.

LP_COLOR_CMAKE_DEBUG: string = $MAGENTA

Color for the *Debug* build type of the CMake section.

See also: *LP_COLOR_CMAKE_RWDI* and *LP_COLOR_CMAKE_RELEASE*.

New in version 2.2.

LP_COLOR_CMAKE_RWDI: string = $BLUE

Color for the *RelWithDebInfo* build type of the CMake section.

See also: *LP_COLOR_CMAKE_DEBUG* and *LP_COLOR_CMAKE_RELEASE*.

New in version 2.2.

LP_COLOR_CMAKE_RELEASE: string = $CYAN

Color for the *Release* build type of the CMake section.

See also: *LP_COLOR_CMAKE_DEBUG* and *LP_COLOR_CMAKE_RWDI*.

New in version 2.2.

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.

May be used by *LP_ENABLE_VCS_REMOTE*.

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* and *LP_MARK_VCS_REMOTE*.

LP_COLOR_CONTAINER: string = $BOLD_BLUE

Color used to indicate that the current shell is running in a container.

New in version 2.1.

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_DISK: string = $BOLD_RED

Color used for displaying information about the hard drive hosting the current directory.

See also *LP_COLOR_DISK_UNITS*, *LP_ENABLE_DISK*, *LP_ALWAYS_DISPLAY_VALUES*, and *LP_PERCENTS_ALWAYS*.

New in version 2.2.

LP_COLOR_DISK_UNITS: string = $RED

Color used for displaying the unit of the available space on the hard drive hosting the current directory.

See also *LP_COLOR_DISK*, *LP_ENABLE_DISK*, *LP_ALWAYS_DISPLAY_VALUES*, and *LP_PERCENTS_ALWAYS*.

New in version 2.2.

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_ENV_VARS_SET: string = $BOLD_BLUE

Color of the environment variables that are set, in the user-defined watch list.

See also: - *LP_ENABLE_ENV_VARS* - *LP_ENV_VARS* - *LP_COLOR_ENV_VARS_UNSET* - *LP_MARK_ENV_VARS_OPEN* - *LP_MARK_ENV_VARS_SEP* - *LP_MARK_ENV_VARS_CLOSE*

LP_COLOR_ENV_VARS_UNSET: string = $BLUE

Color of the environment variables that are unset, in the user-defined watch list.

See also:

- *LP_ENABLE_ENV_VARS*
- *LP_ENV_VARS*
- *LP_COLOR_ENV_VARS_SET*
- *LP_MARK_ENV_VARS_OPEN*
- *LP_MARK_ENV_VARS_SEP*
- *LP_MARK_ENV_VARS_CLOSE*

New in version 2.2.

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_MULTIPLEXER_OPEN* and *LP_MARK_MULTIPLEXER_CLOSE* if the terminal is in a multiplexer.

Changed in version 2.2: Can be disabled by *LP_ENABLE_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_KUBECONTEXT: string = $CYAN

Color used for the current Kubernetes context.

See also: *LP_ENABLE_KUBECONTEXT*.

New in version 2.1.

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_MODULES: string = $BLUE

Color used for displaying currently loaded modules (if *LP_ENABLE_MODULES_HASHCOLOR* is disabled).

See also: *LP_ENABLE_MODULES*.

LP_COLOR_NODE_VENV: string = $LP_COLOR_VIRTUALENV

Color used for displaying a Node.js virtual environment.

See also: *LP_ENABLE_NODE_VENV*.

New in version 2.1.

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_OS_ARCH: string = $MAGENTA

Color used for OS' architecture (e.g. "x86_64", "i686"…).

See also: *LP_ENABLE_OS* and *LP_ENABLE_OS_ARCH*.

New in version 2.2.

LP_COLOR_OS_DISTRIB: string = $MAGENTA

Color used for OS' distribution (e.g. "Ubuntu", "Debian"…).

Note: Will probably only work on Linux-like systems.

See also: *LP_ENABLE_OS* and *LP_ENABLE_OS_DISTRIB*.

New in version 2.2.

`LP_COLOR_OS_FAMILY: string = $MAGENTA`

Color used for OS' family (e.g. "BSD", "GNU"…).

See also: *LP_ENABLE_OS* and *LP_ENABLE_OS_FAMILY*.

New in version 2.2.

`LP_COLOR_OS_KERNEL: string = $MAGENTA`

Color used for OS' kernel (e.g. "Linux", "MinGW"…).

See also: *LP_ENABLE_OS* and *LP_ENABLE_OS_KERNEL*.

New in version 2.2.

`LP_COLOR_OS_VERSION: string = $MAGENTA`

Color used for OS' version codename (e.g. "focal", "buster"…).

Note: Will probably only work on Linux-like systems.

See also: *LP_ENABLE_OS* and *LP_ENABLE_OS_VERSION*.

New in version 2.2.

`LP_COLOR_PATH: string = ""`

Color used for the current working directory.

If *LP_COLOR_PATH_LAST_DIR*, *LP_COLOR_PATH_VCS_ROOT*, *LP_COLOR_PATH_SEPARATOR*, or *LP_COLOR_PATH_SHORTENED* are set, their respective sections will be colored with them instead.

Changed in version 2.0: Default value changed from `$BOLD` to the default color.

`LP_COLOR_PATH_LAST_DIR: string = $BOLD`

Color used for the last path segment, which corresponds to the current directory basename.

New in version 2.0.

`LP_COLOR_PATH_ROOT: string = $BOLD_YELLOW`

Color used in place of *LP_COLOR_PATH* when the current user is root.

`LP_COLOR_PATH_SEPARATOR: string = lp_terminal_format 8 -1 0 0 -1 # Grey`

Color used for the separator ('/') between path segments. If set to the empty string, the separator will take the format of the path segment before it.

`LP_COLOR_PATH_SHORTENED: string = lp_terminal_format 8 -1 0 0 -1 # Grey`

Color used for path segments that have been shortened.

LP_ENABLE_SHORTEN_PATH must be enabled to have any effect.

`LP_COLOR_PATH_VCS_ROOT: string = $BOLD`

Color used for the path segment corresponding to the current VCS repository root directory.

LP_PATH_VCS_ROOT must be enabled to have any effect.

New in version 2.0.

LP_COLOR_PROXY: string = $BOLD_BLUE

Color used for *LP_MARK_PROXY*.

See also: *LP_ENABLE_PROXY*.

LP_COLOR_RAM: string = $BOLD_RED

Color used for displaying information about the available RAM.

See also *LP_COLOR_RAM_UNITS*, *LP_ENABLE_RAM*, *LP_ALWAYS_DISPLAY_VALUES*, and *LP_PERCENTS_ALWAYS*.

New in version 2.2.

LP_COLOR_RAM_UNITS: string = $RED

Color used for displaying the unit of the available RAM.

See also *LP_COLOR_RAM*, *LP_ENABLE_RAM*, *LP_ALWAYS_DISPLAY_VALUES*, and *LP_PERCENTS_ALWAYS*.

New in version 2.2.

LP_COLOR_RUBY_VENV: string = $LP_COLOR_VIRTUALENV

Color used for displaying a Ruby virtual environment.

See also: *LP_ENABLE_RUBY_VENV*.

New in version 2.1.

LP_COLOR_RUNTIME: string = $YELLOW

Color used for displaying the last command runtime.

See also: *LP_ENABLE_RUNTIME*.

LP_COLOR_SHLVL: string = $BOLD_GREEN

Color used for displaying the nested shell level.

See also: *LP_ENABLE_SHLVL* and *LP_MARK_SHLVL*.

New in version 2.1.

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_TERRAFORM: string = $PINK

Color used for displaying a Terraform workspace.

See also: *LP_ENABLE_TERRAFORM*.

New in version 2.1.

`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 environment 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.

See also *`LP_ENABLE_DISPLAY`*.

`LP_COLOR_X11_ON: string = $GREEN`

Color used for indicating that a display is connected.

See also *`LP_ENABLE_DISPLAY`*.

CHAPTER

FIVE

THEMING

Liquid Prompt 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 Liquid Prompt version 1), that allows for custom prompt ordering in the default theme.

If you just want to change icons and/or colors, you can just rely on "presets", which are just regular configuration files. Presets can generally be combined, and some themes may use —or be compatible with— different presets. See *Config Options*.

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

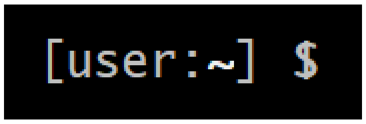
See the Liquid Prompt Theme List on the wiki for user created themes.

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

5.1 Default Theme

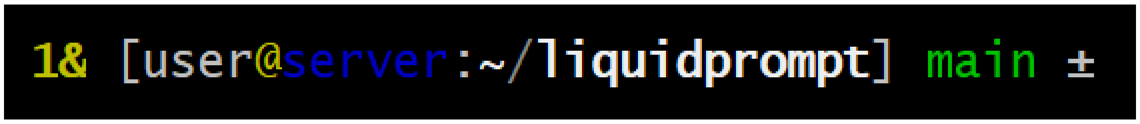
5.1.1 Preview

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



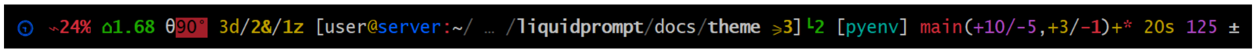
```
[user:~] $
```

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



```
⊙ ⌁24% ⌂1.68 θ90° 3d/2&/1z [user@server:~/ … /liquidprompt/docs/theme ⩾3]└2 [pyenv] main(+10/-5,+3/-1)+* 20s 125 ±
```

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

5.1.2 Configuration

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`*.

5.1.3 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 Liquid Prompt 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: minimal.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, Liquid Prompt 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_WIFI

The lowest wireless signal strength, displayed with an intensity color map as strength decreases. See *LP_ENABLE_WIFI_STRENGTH*.

New in version 2.1.

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*.

New in version 2.0.

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_PROXY

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

LP_ENVVARS

Some user-defined environment variable's states. Watched variables should be defined in the *LP_ENV_VARS* array.

Set variables are displayed in bold blue, unset variables in blue. See also *LP_ENABLE_ENV_VARS*.

New in version 2.2.

LP_SHLVL

The number of nested shells, prefixed with *LP_MARK_SHLVL*, all colored with *LP_COLOR_SHLVL*. Can be disabled by *LP_ENABLE_SHLVL*.

New in version 2.1.

LP_DEV_ENV

Sections related to development environments (see section *dev_env* below).

See also *LP_MARK_DEV_OPEN*, *LP_MARK_DEV_MID*, and *LP_MARK_DEV_CLOSE*.

New in version 2.2.

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_ERR_MEANING

A guess on the meaning of the last error. See *LP_ENABLE_ERROR_MEANING* and *LP_ERR*.

New in version 2.2.

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.

Development environment sections

Version 2.2 of Liquid Prompt introduced a new way to display all sections related to development environments. All the data below are now items in the *LP_DEV_ENV* list, joined by *LP_MARK_DEV_MID* and surrounded by *LP_MARK_DEV_OPEN* and *LP_MARK_DEV_CLOSE*.

If you want to change the ordering or have different marks for different items, you can derive your own theme template and use the following sections.

LP_SCLS

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

LP_AWS_PROFILE

The current active AWS Profile. See *LP_ENABLE_AWS_PROFILE*.

New in version 2.1.

LP_CONTAINER

The container status for the current shell. See *LP_ENABLE_CONTAINER*.

New in version 2.1.

LP_VENV

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

LP_NODE_VENV

The current Node.js virtual environment. See *LP_ENABLE_NODE_VENV*.

New in version 2.1.

LP_RUBY_VENV

The current Ruby virtual environment. See *LP_ENABLE_RUBY_VENV*.

New in version 2.1.

LP_TFSPACE

The current Terraform workspace. See *LP_ENABLE_TERRAFORM*.

New in version 2.1.

LP_KUBECONTEXT

The current Kubernetes context. See *LP_ENABLE_KUBECONTEXT*.

New in version 2.1.

LP_CMAKE

Variables from CMake cache configured in this directory:

- the base name of the compiler (without the path),
- the configured generator (without spaces, some of them shortened),
- the build type.

See *LP_ENABLE_CMAKE*.

New in version 2.2.

5.2 Included Themes

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

5.2.1 Alternate VCS Details Theme

The included `themes/powerline/alternate_vcs.theme` file includes a theme extending the default theme but replacing the VCS details display.

Alternate VCS

The `alternate_vcs` theme is an extension of the default theme.

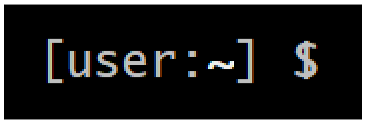
This prompt is a fully usable theme, designed to be more flexible than the default theme in terms of what VCS information is shown in the prompt.

It is also an example of how to build a theme extending the default theme while replacing one of the template sections.

New in version 2.0.

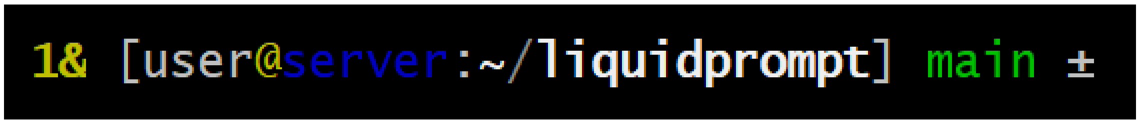
Preview

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



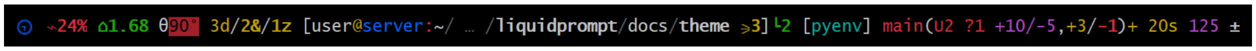
```
[user:~] $
```

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



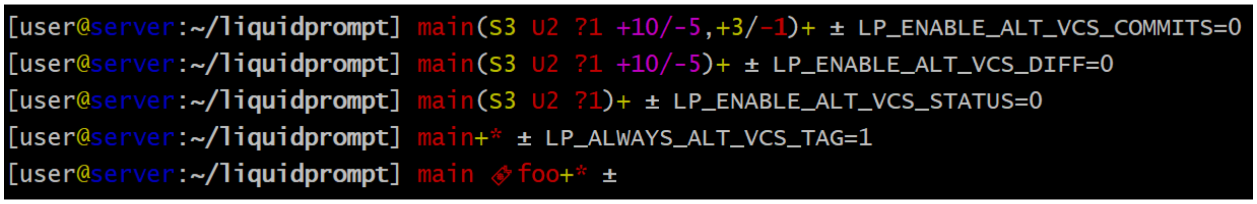
```
1& [user@server:~/liquidprompt] main ±
```

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



```
⊙ ⌁24% ⌂1.68 θ90° 3d/2&/1z [user@server:~/ … /liquidprompt/docs/theme ⩾3] ╰2 [pyenv] main(U2 ?1 +10/-5,+3/-1)+ 20s 125 ±
```

A demo of what disabling the configuration options might look like:



```
[user@server:~/liquidprompt] main(S3 U2 ?1 +10/-5,+3/-1)+ ± LP_ENABLE_ALT_VCS_COMMITS=0
[user@server:~/liquidprompt] main(S3 U2 ?1 +10/-5)+ ± LP_ENABLE_ALT_VCS_DIFF=0
[user@server:~/liquidprompt] main(S3 U2 ?1)+ ± LP_ENABLE_ALT_VCS_STATUS=0
[user@server:~/liquidprompt] main+* ± LP_ALWAYS_ALT_VCS_TAG=1
[user@server:~/liquidprompt] main ◈foo+* ±
```

Configuration

Liquid Prompt Configuration

All Liquid Prompt config options are respected, **except for**:

- *LP_MARK_UNTRACKED* when *LP_ENABLE_ALT_VCS_STATUS* is enabled.

Theme Configuration

Alternate VCS adds these config options:

Features

`LP_ALWAYS_ALT_VCS_TAG: bool = 0`

Determine when a matching VCS tag should be displayed:

- `0` - Only when there is no current branch or bookmark
- 1 - Always

`LP_ENABLE_ALT_VCS_COMMITS: bool = 1`

Display commits ahead or behind the remote tracking branch.

`LP_ENABLE_ALT_VCS_DIFF: bool = 1`

Display the number of changed lines.

`LP_ENABLE_ALT_VCS_STATUS: bool = 1`

Display the number(s) of changed files, of type staged (if VCS supports staging), non-staged (or non-committed if no staging), and untracked.

If disabled, a marker will be added to the end of the display to show if there are untracked files (the behavior of the default theme).

Markers

`LP_MARK_ALT_VCS_TAG: string = ""`

The marker string used to indicate the following string is a VCS tag.

5.2.2 Powerline Theme

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

Powerline

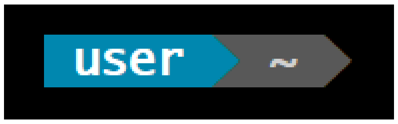
The `powerline` theme is a clone of the Powerline prompt. It copies the default segments of the Powerline prompt for Shell.

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

New in version 2.0.

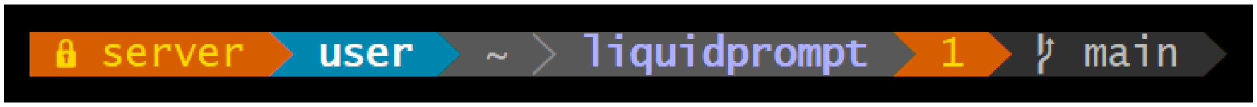
Preview

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



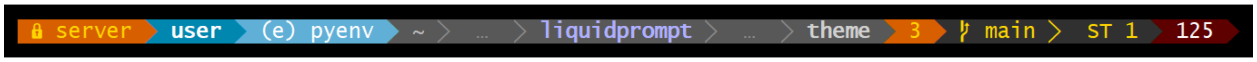
```
user  ~
```

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 Liquid Prompt is displaying nearly everything, it may look like this:



```
🔒 server  user  (e) pyenv  ~ 〉 … 〉 liquidprompt 〉 … 〉 theme  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 Liquid Prompt. 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.

Configuration

Liquid Prompt Configuration

The following Liquid Prompt 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_CHARACTER_KEEP*
- *LP_PATH_KEEP*
- *LP_PATH_LENGTH*
- *LP_PATH_METHOD*
- *LP_PATH_VCS_ROOT*
- *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_PATH_SHORTENED_COLOR: array<int> = (245, 240, 0, 0, 7, 0)

Color for any sections in the current working directory that are shortened to make the path fit in *LP_PATH_LENGTH*.

POWERLINE_PATH_VCS_COLOR: array<int> = (147, 240, 1, 0, 4, 0)

Color for the current working directory segment corresponding to the current VCS repository root directory.

LP_PATH_VCS_ROOT must be enabled to have any effect.

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.

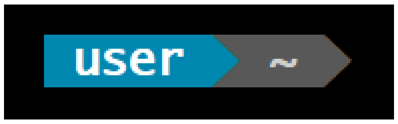
Powerline Full

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

New in version 2.0.

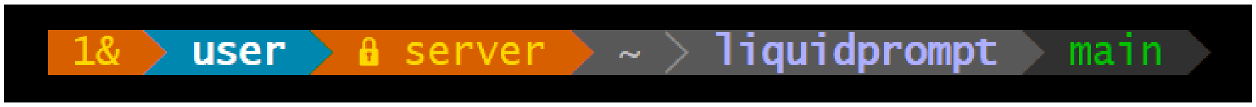
Preview

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



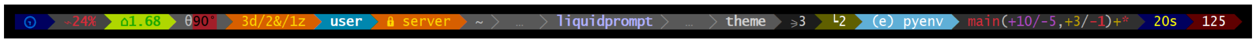
```
user  ~
```

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 Liquid Prompt is displaying nearly everything, it may look like this:



```
⊙  ⌁24%  ⌂1.68  θ90°  3d/2&/1z  user  🔒 server  ~ 〉 … 〉 liquidprompt 〉 … 〉 theme  ⩾3  └2  (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.

Configuration

Liquid Prompt Configuration

All Liquid Prompt config options are respected, **except for**:

- *LP_COLOR_AWS_PROFILE*
- *LP_COLOR_CONTAINER*
- *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_KUBECONTEXT*
- *LP_COLOR_MARK_ROOT*
- *LP_COLOR_MARK_SUDO*
- *LP_COLOR_MARK*
- *LP_COLOR_NODE_VENV*
- *LP_COLOR_NOWRITE*
- *LP_COLOR_PATH_ROOT*
- *LP_COLOR_PATH*
- *LP_COLOR_PROXY*
- *LP_COLOR_RUBY_VENV*
- *LP_COLOR_RUNTIME*
- *LP_COLOR_SHLVL*
- *LP_COLOR_SSH*
- *LP_COLOR_SU*
- *LP_COLOR_TELNET*
- *LP_COLOR_TERRAFORM*
- *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_CLOSE*
- *LP_MARK_BRACKET_OPEN*
- *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_AWS_PROFILE_MARKER: string = "AWS: "

The marker string used to indicate the following string is the name of an AWS profile.

New in version 2.1.

POWERLINE_CHROOT_MARKER: string = "chroot: "

The marker string used to indicate the following string is a chroot.

POWERLINE_KUBECONTEXT_MARKER: string = $LP_MARK_KUBECONTEXT

The marker string used to indicate the following string is the name of a `kubectl` context.

New in version 2.1.

POWERLINE_NODE_ENV_MARKER: string = "node: "

The marker string used to indicate the following string is a Node.js environment.

New in version 2.1.

POWERLINE_PROXY_MARKER: string = "proxy: "

The marker string used to indicate the following string is a HTTP proxy.

POWERLINE_RUBY_ENV_MARKER: string = "ruby: "

The marker string used to indicate the following string is a Ruby environment.

New in version 2.1.

POWERLINE_SOFTWARE_COLLECTION_MARKER: string = "(sc) "

The marker string used to indicate the following string is a Red Hat Software Collection.

POWERLINE_TERRAFORM_ENV_MARKER: string = "(tf) "

The marker string used to indicate the following string is a Terraform workspace.

New in version 2.1.

Colors

POWERLINE_AWS_PROFILE_COLOR: array<int> = (190, 236, 0, 0, 3, 0)

Color for the AWS profile section.

New in version 2.1.

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_CONTAINER_COLOR: array<int> = $POWERLINE_NEUTRAL_COLOR

Color for the container indicator section.

New in version 2.1.

POWERLINE_DIRSTACK_COLOR: array<int> = $POWERLINE_NEUTRAL_COLOR

Color for the directory stack section.

POWERLINE_KUBECONTEXT_COLOR: array<int> = (231, 74, 0, 0, 7, 4)

Color for the Kubernetes context section.

New in version 2.1.

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_NODE_ENV_COLOR: array<int> = $POWERLINE_PYTHON_ENV_COLOR
Color for the Node.js environment section.

New in version 2.1.

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

POWERLINE_RUBY_ENV_COLOR: array<int> = $POWELINE_PYTHON_ENV_COLOR
Color for the Ruby environment section.

New in version 2.1.

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

POWERLINE_SHLVL_COLOR: array<int> = (231, 58, 0, 0, 7, 2)
Color for the nested shell level section.

New in version 2.1.

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_TERRAFORM_ENV_COLOR: array<int> = (231, 182, 0, 0, 7, 4)
Color for the Terraform workspace.

New in version 2.1.

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

POWERLINE_WIFI_STRENGTH_COLOR: array<int> = (-1, 148, 0, 0, -1, 3)
Color for the wireless signal strength section.

New in version 2.1.

5.2.3 Unfold Theme

The included `themes/unfold/unfold.theme` file includes a theme reformatting the default theme to spread it on two lines, with a part of the first line being right-aligned.

Unfold

The `unfold` theme is an reconfiguration of the default theme's structure.

This prompt is a fully usable theme, designed so that:

- The prompt location is more stable (on the second line).
- There is a clearer hierarchy between important data (display on the first line, left side), and less important information (first line, right side).

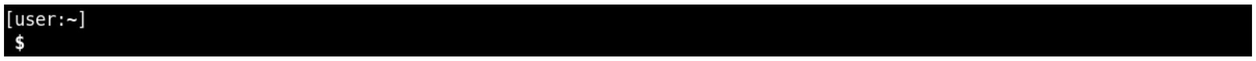
It is also an example of how to rearrange the prompt's template sections, without touching anything else.

As such, all the configuration of the default theme is honored.

New in version 2.2.

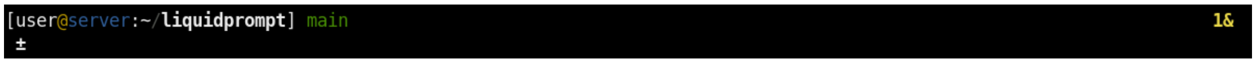
Preview

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



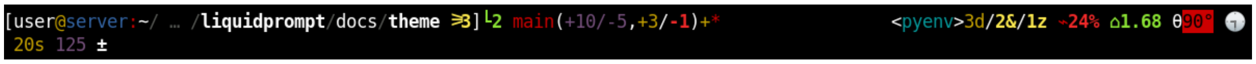
```
[user:~]
 $
```

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



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

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



```
[user@server:~/ … /liquidprompt/docs/theme ⋟3]└2 main(+10/-5,+3/-1)+*          <pyenv>3d/2&/1z ⌁24% ⌂1.68 θ90°
 20s 125 ±
```

5.3 Custom Themes

5.3.1 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.

5.3.2 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.

5.3.3 Examples

The *Alternate VCS Details Theme* is a good example of creating a theme based on the default theme.

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

5.3.4 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.

5.4 Switching Themes

5.4.1 In a nutshell

To use a theme, just source its file in your shell configuration file, after having load the liquidprompt, and then call `lp_theme`:

```
source <your_path>/liquidprompt
source <your_path>/themes/unfold/unfold.theme
lp_theme unfold
```

5.4.2 More Details

Liquid Prompt 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

SIX

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 Liquid Prompt internals. These functions should not be used by users or themes, as they are not guaranteed to not change between versions.

6.1 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*])

Not to be confused with *`_lp_title()`*.

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.

The variable `LP_THEME` can be set to a theme name, which is essentially equivalent to calling this function. `LP_THEME` will always hold the currently active theme; it is updated by this function. If `LP_THEME` is set to an invalid theme name, it will be reset to the previous value.

This function supports shell autocompletion.

New in version 2.0.

Changed in version 2.2: `LP_THEME` reading and setting added.

`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.

For example, to set the error color to a bright, bold pink, with a fallback color of red:

```
lp_terminal_format 204 -1 1 0 1
LP_COLOR_ERR=$lp_terminal_format
```

To set the prompt mark color to black on a white background:

```
lp_terminal_format 0 7
LP_COLOR_MARK=$lp_terminal_format
```

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.

6.2 Data Functions

6.2.1 Version Control Data Functions

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, var:lp_vcs_dir, var:lp_vcs_specific_dir, var:lp_vcs_subtype

Returns `true` if the current directory is part of a version control repository. If not, returns 1. Returns the VCS type ID, subtype if one exists, the VCS data directory, 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.

Git worktrees have two Git directories: one is the main directory in *lp_vcs_dir*, which holds the DB and config. The other is the worktree specific directory in *lp_vcs_specific_dir*, which holds working directory specific information, like index and status like merge or rebase in progress.

Note: *lp_vcs_dir* will not be set for Fossil repositories. Protect it with `"${lp_vcs_dir-}"`.

Note: *lp_vcs_specific dir* will only be set for Git repositories. Protect it with `"${lp_vcs_specific_dir-}"`.

Note: *lp_vcs_subtype* will not be set usually. The only currently supported subtypes are *vcsh* and *svn*, which are subtypes of *git*.

New in version 2.0.

Changed in version 2.1: Added the *lp_vcs_dir* and *lp_vcs_subtype* return values. Added support for checking the `GIT_DIR` environment variable.

Changed in version 2.2: Added the *lp_vcs_specific_dir* return value.

_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 Liquid Prompt 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: Unless otherwise documented, the following functions return `0` for good data, `1` for no data, and `2` 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 repository 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_remote()` → var:lp_vcs_remote

Return `true` if the current branch is a remote tracking branch. The remote name is returned in *lp_vcs_remote*.

Many VCS providers do not have such information. Currently this is only implemented for Git.

Can be enabled by *LP_ENABLE_VCS_REMOTE*.

New in version 2.2.

`_lp_vcs_staged_files()` → var:lp_vcs_staged_files

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

Many VCS providers do not support staging.

New in version 2.0.

`_lp_vcs_staged_lines()` → var:lp_vcs_staged_i_lines, var:lp_vcs_staged_d_lines

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

Many VCS providers do not support staging.

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_i_lines, var:lp_vcs_uncommitted_d_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_i_lines, var:lp_vcs_unstaged_d_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 Liquid Prompt 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_i_lines, var:lp_vcs_uncommitted_d_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 check-in 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 Liquid Prompt 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_i_lines, var:lp_vcs_uncommitted_d_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 Liquid Prompt 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_staged_files``()` → var:lp_vcs_staged_files

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

New in version 2.0.

`_lp_git_staged_lines``()` → var:lp_vcs_staged_i_lines, var:lp_vcs_staged_d_lines

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

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_i_lines, var:lp_vcs_uncommitted_d_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_i_lines, var:lp_vcs_unstaged_d_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 Liquid Prompt 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 `2` (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_i_lines, var:lp_vcs_uncommitted_d_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 Liquid Prompt 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_i_lines, var:lp_vcs_uncommitted_d_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.

6.2.2 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`*.

Changed in version 2.1: Implemented *sysfs* method as the default way of getting battery status.

6.2.3 Development Environment

`_lp_cmake()` → var:lp_cmake_compiler, var:lp_cmake_generator, var:lp_cmake_buildtype

Returns `true` if a CMake context is found. Parse the data in *CMakeCache.txt* and returns the basename of the configured compiler, generator (e.g. "Unix Makefiles"), and build type ("Debug", "Release", etc.). Some generator names are shorten: "Makefiles" becomes "Make" and "Visual Studio" becomes "VS".

Can be disabled by *`LP_ENABLE_CMAKE`*.

New in version 2.2.

`_lp_kubernetes_context()` → var:lp_kubernetes_context, var:lp_kubernetes_namespace

Returns `true` if a Kubernetes context is found. Returns the Kubernetes context name or the first name component.

Splitting long context names into components is defined by *`LP_DELIMITER_KUBECONTEXT_SUFFIX`* and *`LP_DELIMITER_KUBECONTEXT_PREFIX`*. Both use greedy matches - see *Config Options* for examples.

If enabled by *`LP_ENABLE_KUBE_NAMESPACE`*, will also return the default namespace for the current context, if one is set.

Can be disabled by *`LP_ENABLE_KUBECONTEXT`*.

New in version 2.1.

`_lp_node_env()` → var:lp_node_env

Returns `true` if a Node.js environment is detected. Returns the virtual environment name.

Can be enabled by *`LP_ENABLE_NODE_VENV`*.

New in version 2.1.

`_lp_python_env()` → var:lp_python_env

Returns `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.

Changed in version 2.1: Displays the "prompt string" first (the `--prompt` argument when setting up the virtualenv).

`_lp_ruby_env()` → var:lp_ruby_env

Returns `true` if a RVM or RBENV ruby environment is detected. Returns the virtual environment name.

In the case of a RVM environment, the label displayed can be customized with the *LP_RUBY_RVM_PROMPT_OPTIONS*.

Can be disabled by *LP_ENABLE_RUBY_VENV*.

New in version 2.1.

`_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.

`_lp_terraform_env()` → var:lp_terraform_env

Returns `true` if a Terraform workspace is detected. Returns the workspace name.

Can be enabled by *LP_ENABLE_TERRAFORM*.

New in version 2.1.

6.2.4 Disks and Memory

`_lp_disk -> var:lp_disk, var:lp_disk_human, var:lp_disk_perc`

Gather information about the current state of the hard drive hosting the *current directory*:

- available space in kibi-bytes (`lp_disk`, that is, 1024 bytes),
- available space in human-readable form, using binary unit prefixes (`lp_disk_human`, see also *__lp_bytes_to_human()*).
- available space as a percentage of total (`lp_disk_perc`).

Returns `true` if the used space is below at least one of the user-defined thresholds:

- *LP_DISK_THRESHOLD*
- *LP_DISK_THRESHOLD_PERC*

Can be disabled by *LP_ENABLE_DISK*.

New in version 2.2.

`_lp_ram -> var:lp_ram, var:lp_ram_human, var:lp_ram_perc`

Gather information about the current state of the RAM:

- available space in kibi-bytes (`lp_ram`, that is, 1024 bytes),
- available space in human-readable form, using binary unit prefixes (`lp_ram_human`, see also *__lp_bytes_to_human()*).
- available space as a percentage of total (`lp_ram_perc`).

Returns `true` if the used space is below at least one of the user-defined thresholds:

- *LP_RAM_THRESHOLD*
- *LP_RAM_THRESHOLD_PERC*

Can be disabled by *LP_ENABLE_RAM*.

New in version 2.2.

6.2.5 Environment

`_lp_aws_profile``()` → var:lp_aws_profile

Returns `true` if the `AWS_PROFILE`, `AWS_DEFAULT_PROFILE`, or `AWS_VAULT` variables are found in the environment (in that order of preference). Returns the contents of the variable.

Can be disabled by *LP_ENABLE_AWS_PROFILE*.

New in version 2.1.

`_lp_connected_display``()`

Returns `true` if there is a connected X11 display.

New in version 2.0.

Changed in version 2.2: Can be disabled by *LP_ENABLE_DISPLAY*.

`_lp_connection``()` → var:lp_connection

Returns a string matching the connection context of the shell. Valid values:

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

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_container``()` → var:lp_container

Returns `true` if the shell is running in a container. In that case, the return variable is set to a string matching the container type. Possible values include (but are not limited to):

- `Singlrty` - running in a Singularity container
- `Toolbox` - running in a Toolbox container
- `Podman` - running in a Podman container
- `Docker` - running in a Docker container
- `LXC` - running in an LXC container
- `nspawn` - running in a systemd-nspawn container

It is not possible to detect more than one containerization type to be returned. The containers are checked in the order listed above, and the first one found will be returned.

Can be enabled by *LP_ENABLE_CONTAINER*.

New in version 2.1.

_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_env_vars([*color_if_set*[, *color_if_unset*[, *end_color*]]]) → var:lp_env_vars

Gather the states of the environment variables indicated in the *LP_ENV_VARS* array, and put them in the `lp_env_vars` array.

LP_ENV_VARS should be a list of environment variable names to look for, along with the string to be displayed if the variable is set, and an optional string to be displayed if the variable is not set. The string to be displayed may contain a `%s` marker, which will be replaced by the variable's content.

If `color_if_set` is passed, it will be used to color the *set* variables string. If `color_if_unset` is passed, it will be used to color the *unset* variables string.

`end_color` is added at the end of each variable string. It defaults to "$NO_COL" (color reset).

Returns `true` if at least one variable representation is added to the result array. Returns `1` if the no variable representation is set. Returns `2` if the user disabled the feature with *LP_ENABLE_ENV_VARS*.

New in version 2.2.

_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_error_meaning() → var:lp_error_meaning

Returns `true` if the last user shell command returned a non-zero exit code. Returns (in the return variable) a guess of the meaning of that error.

Can be disabled by *LP_ENABLE_ERROR_MEANING*.

New in version 2.2.

_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_multiplexer

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.

Changed in version 2.2: Can be disabled by *LP_ENABLE_MULTIPLEXER*, except if `--internal` is passed (i.e. for internal use only).

`_lp_shell_level()` → var:lp_shell_level

Returns `true` if the shell is a nested shell inside another shell.

Can be disabled by *LP_ENABLE_SHLVL*.

New in version 2.1.

`_lp_terminal_device()` → var:lp_terminal_device

Returns the basename of the terminal device connected to the shell's standard input.

Note: This value should never change during the life of the shell.

Note: This data source is unlikely to be wanted by the user, and should not be included in themes by default.

New in version 2.0.

6.2.6 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.

6.2.7 Load

`_lp_cpu_load()` → var:lp_cpu_load

Returns a string of the system load average smallest increment, usually 1 minute. The return code is not defined.

`_lp_load()` → var:lp_load, var:lp_load_adjusted

Returns `true` if the system load average scaled by CPU count is greater than the threshold. Returns the system load average in *lp_load*, and the average scaled by CPU count, multiplied by 100 in *lp_load_adjusted*. In other words, the load average is multiplied by 100, then divided by the number of CPU cores.

lp_load should be displayed to the user, while *lp_load_adjusted* should be used to compare values between machines using *LP_LOAD_CAP*. The default theme uses this to generate a color scale.

Note: *LP_LOAD_CAP* is a raw floating point configuration value that is difficult to do math on. `_LP_LOAD_CAP` contains the same value, but multiplied by 100 to make comparisons to *lp_load_adjusted* simple. Use it along with *lp_load_adjusted* as arguments to *_lp_color_map()*.

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.

6.2.8 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.

Changed in version 2.2: Can be disabled by *LP_ENABLE_CHROOT*.

`_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 in *lp_hostname*.

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

New in version 2.0.

Changed in version 2.1: Returns the actual hostname instead of a shell prompt escape code. No longer sets `LP_HOST_SYMBOL` to the same return string. Added *LP_HOSTNAME_METHOD* to configure display method.

`_lp_os``()` → var:lp_os_arch, var:lp_os_family, var:lp_os_kernel, var:lp_os_distrib, var:lp_os_version

Gather data about the current Operating System.

Returns `true` if it was able to gather all possible data. Returns `1` if some expected information was missing. Returns `2` if the user disabled the feature with *LP_ENABLE_OS*.

Returns data in `lp_os_*` variables:

- processor architecture (e.g. x86_64, i686, etc.),
- OS family (BSD, UNIX, GNU or Windows),
- OS kernel (Linux, Darwin, Cygwin, etc.),
- Linux *distribution* (e.g. ubuntu, arch, mandrake, etc.),
- Linux distribution *version codename* (e.g. focal, ada, buzz, etc.)

Each data source can be disabled via its corresponding configuration variable:

- *LP_ENABLE_OS_ARCH*
- *LP_ENABLE_OS_FAMILY*
- *LP_ENABLE_OS_KERNEL*
- *LP_ENABLE_OS_DISTRIB*
- *LP_ENABLE_OS_VERSION*

New in version 2.2.

`_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.

Changed in version 2.1: If the user has NOPASSWD powers, that is cached on startup to prevent multiple `sudo` calls.

`_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 in *lp_username*.

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

New in version 2.0.

Changed in version 2.1: Returns the actual username instead of a shell prompt escape code.

6.2.9 Path

`_lp_path_format``(`*path_format=$LP_COLOR_PATH*, *last_directory_format=$path_format*, *vcs_root_format=$last_directory_format*, *shortened_directory_format=$path_format*, *separator="/"*[, *separator_format*]`)` → var:lp_path, var:lp_path_format

Returns a shortened and formatted string indicating the current working directory path. *lp_path* contains the path without any formatting, custom separators, or shell escapes, intended for use in the terminal title. *lp_path_format* contains the complete formatted path, to be inserted into the prompt.

The behavior of the shortening is controlled by *`LP_ENABLE_SHORTEN_PATH`*, *`LP_PATH_METHOD`*, *`LP_PATH_LENGTH`*, *`LP_PATH_KEEP`*, *`LP_PATH_CHARACTER_KEEP`*, and *`LP_PATH_VCS_ROOT`*. See their descriptions for details on how they change the output of this function.

The last directory in the displayed path will be shown with the *last_directory_format*.

If a VCS repository is detected with *`_lp_find_vcs()`*, the root of the VCS repository is formatted with *vcs_root_format*. The detection method is the same as for all other VCS display, so if a VCS type or directory is disabled, it will not be detected.

If the path shortening shortens a directory (or multiple consecutive directories), they will be formatted with *shortened_directory_format*.

A custom *separator* will only be substituted in the *lp_path_format* output. Note that this will cut into maximum path length if the separator is longer than one character.

With no specified *separator_format*, each separator will take the format of the directory section preceding it. Otherwise every separator will be formatted with *separator_format*. Note that the root directory is treated as a directory, and is formatted as such.

New in version 2.0.

Changed in version 2.1: Changed *lp_path* to no longer contain shell escapes.

6.2.10 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.

6.2.11 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.

If no temperature data is found, returns `false` and *lp_temperature* will not be set.

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`.

6.2.12 Time

`_lp_time``()` → var:lp_time

Returns `true` if digital time is enabled. Returns the current digital time string, formatting set by *`LP_TIME_FORMAT`*.

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

New in version 2.0.

Changed in version 2.1: Returns the actual time instead of a shell prompt escape code.

`_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.

6.2.13 Wireless

`_lp_wifi_signal_strength``()` → var:lp_wifi_signal_strength

Returns `true` if the lowest wireless signal strength is lower than the threshold. Returns the lowest strength percentage.

If the threshold is not surpassed, the lowest signal strength is still returned.

If no wireless signal data is found, returns `false` and *lp_wifi_signal_strength* will not be set.

The threshold is configured with *LP_WIFI_STRENGTH_THRESHOLD*.

Can be disabled by *LP_ENABLE_WIFI_STRENGTH*.

New in version 2.1.

6.3 Default Theme 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.

6.3.1 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 *_lp_default_theme_prompt_data()* then *_lp_default_theme_prompt_template()*.

New in version 2.0.

_lp_default_theme_prompt_data()

Runs all of the below theme data functions, and writes values to the *Default Theme* variables. Can be used to generate all the default theme sections, then modify them before running a user template.

New in version 2.0.

_lp_default_theme_prompt_template()

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. Can be used to set different template sections than the default theme, but still use the same template engine.

New in version 2.0.

6.3.2 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_aws_profile_color() → var:lp_aws_profile_color

Returns *_lp_aws_profile()* with color from *LP_COLOR_AWS_PROFILE*.

New in version 2.1.

Changed in version 2.2: No longer include squared brackets, superseded by *LP_MARK_DEV_OPEN*, *LP_MARK_DEV_MID* and *LP_MARK_DEV_CLOSE*.

_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_cmake_color() → var:lp_cmake_color

Returns data from *_lp_cmake()*. *lp_cmake_compiler* and *lp_cmake_generator* are colored according to their hash (see *_lp_hash_color()*). *lp_cmake_buildtype* has as configurable color, depending on its value:

- *Debug*, colored with *LP_COLOR_CMAKE_DEBUG* (magenta, by default),
- *RelWithDebInfo*, colored with *LP_COLOR_CMAKE_RWDI* (blue, by default),
- *Release*, colored with *LP_COLOR_CMAKE_RELEASE* (cyan, by default),
- any other value would be colored according to its hash.

New in version 2.2.

_lp_container_color() → var:_lp_container_color

Returns *_lp_container()*, surrounded by « and » colored with *LP_COLOR_CONTAINER* if the value is true. Returns no data if the value is false.

New in version 2.1.

Changed in version 2.2: No longer include squared brackets, superseded by *LP_MARK_DEV_OPEN*, *LP_MARK_DEV_MID* and *LP_MARK_DEV_CLOSE*.

_lp_dev_env_color() → var:lp_dev_env_color

Assemble data related to development environment and returns a single string. The returned string starts with *LP_MARK_DEV_OPEN* and ends with *LP_MARK_DEV_CLOSE*, with each section separated by *LP_MARK_DEV_MID*.

Data are collected from:

- *LP_SCLS*
- *LP_AWS_PROFILE*
- *LP_CONTAINER*
- *LP_VENV*
- *LP_NODE_VENV*
- *LP_RUBY_VENV*
- *LP_TFSPACE*
- *LP_KUBECONTEXT*
- *LP_CMAKE*

New in version 2.2.

_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_disk_color() → var:lp_disk_color

Returns information about available space of the hard drive hosting the current directory.

If *LP_ALWAYS_DISPLAY_VALUES* is `false`, displays a colored mark (using *LP_MARK_DISK*), if the available disk space goes below *LP_DISK_THRESHOLD* or *LP_DISK_THRESHOLD_PERC*. If it is `true`, displays the corresponding value, either as a percentage (if *LP_DISPLAY_VALUES_AS_PERCENTS* is `true`) or as a human-readable quantity (if *LP_DISPLAY_VALUES_AS_PERCENTS* is `false`).

The mark and the value itself are colored with *LP_COLOR_DISK*, while the unit is colored with *LP_COLOR_DISK_UNITS*.

New in version 2.2.

_lp_env_vars_color() → var:lp_env_vars_color

Returns the elements of the array set by *_lp_env_vars()*, joined with the *LP_MARK_ENV_VARS_SEP* marker, and surrounded by *LP_MARK_ENV_VARS_OPEN* and *LP_MARK_ENV_VARS_CLOSE*.

If a matching environment variable is set, it is colored with *LP_COLOR_ENV_VARS_SET*, if it is unset, it is colored with *LP_COLOR_ENV_VARS_UNSET*.

See also *LP_ENV_VARS*.

New in version 2.2.

_lp_error_color() → var:lp_error_color

Returns *_lp_error()* with color from *LP_COLOR_ERR*.

New in version 2.0.

_lp_error_meaning_color() → var:lp_error_meaning_color

Returns *_lp_error_meaning()* with color from *LP_COLOR_ERR* and surrounded by parentheses.

New in version 2.2.

_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_kubernetes_context_color() → var:lp_kubernetes_context_color

Returns data from *_lp_kubernetes_context()*, colored with *LP_COLOR_KUBECONTEXT* and using mark *LP_MARK_KUBECONTEXT*.

New in version 2.1.

Changed in version 2.2: No longer include squared brackets, superseded by *LP_MARK_DEV_OPEN*, *LP_MARK_DEV_MID* and *LP_MARK_DEV_CLOSE*.

_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_node_env_color() → var:lp_node_env_color

Returns *_lp_node_env()* with color from *LP_COLOR_NODE_VENV*.

New in version 2.1.

Changed in version 2.2: No longer include squared brackets, superseded by *LP_MARK_DEV_OPEN*, *LP_MARK_DEV_MID* and *LP_MARK_DEV_CLOSE*.

_lp_os_color() → var:lp_os_color

Gather information about the Operating System.

Any string encountered in the process may be replaced by a user-defined counterpart, using the *LP_MARK_OS* configuration array.

If the string was not replaced, it is colored with a random color depending on its hash (see *_lp_hash_color()*).

All fields gathered via the *_lp_os()* function are joined with the *LP_MARK_OS_SEP* string, in the following order: arch, family, kernel, distribution, codename. The corresponding data are returned as a single string via the `lp_os_color` variable.

The function returns 2 if the user disabled the feature with *LP_ENABLE_OS*, 1 if no field was filled in with data, and `true` otherwise.

New in version 2.2.

_lp_python_env_color() → var:lp_python_env_color

Returns *_lp_python_env()* with color from *LP_COLOR_VIRTUALENV*.

New in version 2.0.

Changed in version 2.2: No longer include squared brackets, superseded by *LP_MARK_DEV_OPEN*, *LP_MARK_DEV_MID* and *LP_MARK_DEV_CLOSE*.

_lp_ram_color() → var:lp_ram_color

Returns information about available RAM.

If *LP_ALWAYS_DISPLAY_VALUES* is `false`, displays a colored mark (using *LP_MARK_RAM*), if the available ram goes below *LP_RAM_THRESHOLD* or *LP_RAM_THRESHOLD_PERC*. If it is `true`, displays the corresponding value, either as a percentage (if *LP_DISPLAY_VALUES_AS_PERCENTS* is `true`) or as a human-readable quantity (if *LP_DISPLAY_VALUES_AS_PERCENTS* is `false`).

The mark and the value itself are colored with *LP_COLOR_RAM*, while the unit is colored with *LP_COLOR_RAM_UNITS*.

New in version 2.2.

_lp_ruby_env_color() → var:lp_ruby_env_color

Returns *_lp_ruby_env()* with color from *LP_COLOR_RUBY_VENV*.

New in version 2.1.

Changed in version 2.2: No longer include squared brackets, superseded by *LP_MARK_DEV_OPEN*, *LP_MARK_DEV_MID* and *LP_MARK_DEV_CLOSE*.

_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_shell_level_color() → var:lp_shell_level_color

Returns *_lp_shell_level()*, prefixed with *LP_MARK_SHLVL*, all colored with *LP_COLOR_SHLVL*.

New in version 2.1.

_lp_software_collections_color() → var:lp_software_collections_color

Returns *_lp_software_collections()* with color from *LP_COLOR_VIRTUALENV*.

New in version 2.0.

Changed in version 2.2: No longer include squared brackets, superseded by *LP_MARK_DEV_OPEN*, *LP_MARK_DEV_MID* and *LP_MARK_DEV_CLOSE*.

_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_terraform_env_color() → var:lp_terraform_env_color

Returns *_lp_terraform_env()* with color from *LP_COLOR_TERRAFORM*.

New in version 2.1.

Changed in version 2.2: No longer include squared brackets, superseded by *LP_MARK_DEV_OPEN*, *LP_MARK_DEV_MID* and *LP_MARK_DEV_CLOSE*.

_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.

_lp_wifi_signal_strength_color() → var:lp_wifi_signal_strength_color

Returns *_lp_wifi_signal_strength()* with color from *LP_COLORMAP* and mark from *LP_MARK_WIFI*.

New in version 2.1.

6.4 Utility Functions

These functions are designed to be used by themes.

_lp_as_text(*string*) → stdout

Deprecated since version 2.1: Use *__lp_strip_escapes()* instead.

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_bytes_to_human(*bytes*[, *precision*]) → var:lp_bytes, var:lp_bytes_units

Convert the given bytes into a human readable format, using binary multiple-byte units. E.g.: KiB instead of KB, 1024 instead of 1000.

Precision can be 0, 1 or 2 digits. If not given, precision is two digits.

Converted value goes into `lp_bytes` and its unit in `lp_bytes_units`.

Note that after petabytes (PiB) of data, Bash and Zsh will start failing at integer computations.

New in version 2.2.

_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_create_link(*url*, *text*) → var:lp_link

Adds the *url* link to the given *text*.

See *LP_ENABLE_HYPERLINKS*.

`_lp_create_link_path`(*path*) → lp_link_path

Adds a link on the given path, with the protocol scheme depending on the current connection type.

If the current connection is *SSH*, adds an `SFTP://` URL, if it is *su* or *lcl* (see *`_lp_connection()`*), adds a `file://` one.

See also *`LP_ENABLE_HYPERLINKS`* and *`_lp_create_link()`*.

`_lp_fill`(*left*, *right*[, *fillstring*[, *splitends*]]) → var:lp_fill

Adds as much *fillstring* (e.g. spaces) between *left* and *right*, so as to make the resulting string the same width as the current terminal.

If *fillstring* is omitted, it defaults to one space.

If *fillstring* is a string with several characters and *splitends* is 1 (the default), then the final occurrence of *fillstring* will have its end cut, so as to fit the terminal width.

If *fillstring* has multiple characters and *splitends* is 0, some spaces will be inserted after the last occurrence of *fillstring*, so as to match the exact width of the terminal.

Note:

Any escaped sequence in *fillstring* will be removed automatically.
The end of *left* and the beginning of *right* may be used to add escaped sequences at the beginning and, respectively, the end of the filling sequence.

If the available number of columns in the terminal is smaller than the width of *left* and *right* combined, then the function will return code 1 and set *lp_fill* to *left* and *right*, concatenated.

For example, `_lp_fill "Left part·" "·right part" "" 1` will render (in a terminal being 32 characters large):

Left part··right part

New in version 2.2.

`_lp_formatted_title`(*title*)

Sets the theme generated title to *title*. The input is escaped using *`__lp_strip_escapes()`* 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_grep_fields`(*filename*, *delimiter*, *keys...*) → var:lp_grep_fields

Parse the given filename for one key/value pairs of the form "<key><delimiter><value>" (e.g. "this=that") on each line. Sets an array containing the parsed values, for each key in the same order the function was called.

Listing 1: Example of use

```
_lp_grep_fields "CMakeCache.txt" "=" "CMAKE_C_COMPILER:FILEPATH" "CMAKE_CXX_
↪COMPILER:FILEPATH"
cmake_c_compiler=${lp_grep_fields[_LP_FIRST_INDEX+0]-}
cmake_cxx_compiler=${lp_grep_fields[_LP_FIRST_INDEX+1]-}
```

Note: Bash and Zsh are using different array indexing schemes. To write portable code, you should use `_LP_FIRST_INDEX`.

> **Warning:** It is strongly advised not to loop over the items in *lp_grep_fields*. If a searched key is missing in the file, its corresponding entry in the array will be silently skipped, and thus the indices you would expect may lead to unset variables. Just use explicit indexing to access the parsed values.

Returns 1 if the file does not exists.

New in version 2.2.

`_lp_hash_color`(*str*) → var:lp_hash_color

Colorize the given string with a color depending on its hash. The color is chosen among: (green, yellow, blue, purple, cyan). Note that the red color is not a candidate, as it should be reserved for alerts.

New in version 2.2.

`_lp_join`(*delimiter*, *items...*) → var:lp_join

Join all strings in items with the given delimiter. Example: `_lp_join ", " "a" "b" "c"` will render `lp_join="a, b, c"`

New in version 2.2.

`_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_formatted_title()`* instead.

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

New in version 2.0.

`_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*][, *vcs_subtype*]) → 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", matches *vcs_subtype* to see if the repository is of type VCSH or `git-svn` instead, and return their marks if so. If *vcs_subtype* is not set, uses the value of `lp_vcs_subtype` instead.

Changed in version 2.1: Added *vcs_subtype* argument.

`_lp_substitute`(*string*, *pairs_array*) → var:lp_substitute

If the given string is found in the given array of pairs, return the second element of the pair for which the first element matches the string.

For instance:

```
pairs=(
    "A" "B" # Replace A by B.
    "something" "dead pixels"
    "I see" "nothing"
)
_lp_substitute "something" "${pairs[@]}"
# "$lp_substitute" == "dead pixels"
```

New in version 2.2.

_lp_title(*title*) → stdout

Not to be confused with *lp_title()*.

Deprecated since version 2.0: Use *_lp_formatted_title* instead.

Formats *title* with title escape codes. The input is escaped using *__lp_strip_escapes()* 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_version_greatereq(*major*, *minor*[, *patch*[, *string*[, *number*]]])

Returns true (0) if Liquid Prompt version is greater than or equal to the the given version. Returns 1 (false) if there is a *minor* or less version difference, and 2 (false) if it is a *major* difference.

See also *_lp_version_string()*.

Warning: This only supports the following input values for *strings*: “alpha”, “beta” and “rc”.

New in version 2.2.

_lp_version_string([*major*[, *minor*[, *patch*[, *string*[, *number*]]]]]) → var:lp_version

Formats the given version number in a version string of the form: “${major}.${minor}.${patch}-${string}.${number}”

If no version is given, formats the current version number of Liquid Prompt. If a version number is given, *major* and *minor* are both mandatory.

See also *_lp_version_greatereq()*.

New in version 2.2.

6.5 Internal Functions

These functions are designed to be used only by Liquid Prompt 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 Liquid Prompt.

6.5.1 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.

6.5.2 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`.

6.5.3 Battery

`__lp_battery_acpi``()` → var:lp_battery, var:lp_battery_status

Returns the status and remaining capacity of the battery, as reported by the *acpi* tool. This function is available only on Linux, and requires *acpi* to be installed.

New in version 2.1.

`__lp_battery_sysfs``()` → var:lp_battery, var:lp_battery_status

Returns the status and remaining capacity of the battery, using *sysfs*. This is the default method. If multiple batteries are present, returns the status of the first battery found. This function is available only on Linux systems.

New in version 2.1.

6.5.4 Git

`__lp_git_diff_shortstat_files``(`*diff_shortstat*`)` → var:lp_git_diff_shortstat_files

Processes the input *diff_shortstat* as the output of a `git diff --shortstat` command, returning the number of changed files. This allows for the comparison of any two states, as *`__lp_git_diff_shortstat_files()`* does not run any specific `git diff` command.

New in version 2.0.

`__lp_git_diff_shortstat_lines``(`*diff_shortstat*`)` → var:lp_git_diff_shortstat_lines

Processes the input *diff_shortstat* as the output of a `git diff --shortstat` command, returning the number of changed lines. This allows for the comparison of any two states, as *`__lp_git_diff_shortstat_files()`* does not run any specific `git diff` command.

New in version 2.0.

`__lp_git_diff_shortstat_staged``()` → var:_lp_git_diff_shortstat_staged

Returns the output of a `git diff --shortstat` command, comparing the staging area to the HEAD commit.

The return variable is supposed to be a cache, set as local in *`__lp_set_prompt()`*, preventing duplicate calls to `git`.

New in version 2.0.

`__lp_git_diff_shortstat_uncommitted``()` → var:_lp_git_diff_shortstat_uncommitted

Returns the output of a `git diff --shortstat` command, comparing the working directory to the HEAD commit.

The return variable is supposed to be a cache, set as local in *`__lp_set_prompt()`*, preventing duplicate calls to `git`.

New in version 2.0.

`__lp_git_diff_shortstat_unstaged``()` → var:_lp_git_diff_shortstat_unstaged

Returns the output of a `git diff --shortstat` command, comparing the working directory to the staging area.

The return variable is supposed to be a cache, set as local in *`__lp_set_prompt()`*, preventing duplicate calls to `git`.

New in version 2.0.

6.5.5 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.

6.5.6 OS

`__lp_hostname_hash``()` → var:lp_hostname_hash

Returns the hash of the hostname as computed by `cksum`.

New in version 2.0.

6.5.7 Path

`__lp_end_path_left_shortening``()`

Terminate a multi-directory shortening, checking if the shortening actually made a shorter path, and if so, adding the shortened mark. If not, adds the real path to the output. Only used internally by *`_lp_path_format()`*.

New in version 2.0.

`__lp_get_unique_directory``(`*path*`)` → var:lp_unique_directory

Returns the shortest unique directory prefix matching the real directory input. Only used internally by *`_lp_path_format()`*.

New in version 2.0.

`__lp_pwd_tilde``([`*path*`])` → var:lp_pwd_tilde

Returns *path*, or `PWD` if *path* is not set, with the user's home directory replaced with a tilde ("~").

Changed in version 2.0: Renamed from `_lp_get_home_tilde_collapsed`. Return method changed from stdout. Optional parameter *path* added.

6.5.8 Prompt

`__lp_before_command``()`

Used only by Bash to hack the DEBUG trap to run functions before the user command executes.

Changed in version 2.1: Renamed from the Bash version of `__lp_runtime_before`.

`__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`*, track current directory changes, and initialize data source cache variables. This function also calls the current theme functions.

Changed in version 2.0: Renamed from `_lp_set_prompt`.

6.5.9 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``()`

Called by *`__lp_set_prompt()`* 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`.

6.5.10 Theme

`__lp_theme_list``()` → var:lp_theme_list

Returns an array of Liquid Prompt 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.

6.5.11 Title

`__lp_get_last_command_line``()` → var:command

Returns the whole command line most recently submitted by the user.

New in version 2.1.

`__lp_print_title_command``()`

Sets the terminal title to the normal set title, postpended with the currently running command.

New in version 2.1.

6.5.12 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`.

Changed in version 2.1: No longer takes arguments of what backends to try.

`__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`.

`__lp_temp_sysfs``()` → var:lp_temperature

A temperature backend reading directly from the Linux sysfs file system.

New in version 2.1.

6.5.13 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_floating_scale``(`*number*, *scale*`)` → var:ret

Returns the input floating point *number* multiplied by the input *scale*. The input *scale* must be a power of 10.

Shells do not support floating point math, so this is used to scale up floating point numbers to integers with the needed precision.

New in version 2.0.

`__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.

`__lp_strip_escapes``(`*string*`)` → var:ret

Remove shell escape characters so *string* prints correctly in a terminal title, or can be measured for printing character length.

New in version 2.1.

6.5.14 Wireless

`__lp_wifi_signal_strength_raw()` → var:level

Returns the lowest raw wireless signal strength in dBm. Return `2` if no data is found.

Implementation depends on operating system. This function does not exist on all operating systems.

New in version 2.1.

CHAPTER

SEVEN

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 Liquid Prompt*.

7.1 Version 2.1 Release Notes

Version 2.1 has a whole host of new data sources, features, and compatibility fixes.

7.1.1 Title Command

Liquid Prompt can now display the currently running command as part of the terminal title. See *LP_ENABLE_TITLE_COMMAND* for more information.

7.1.2 Development Environments

Multiple new development environments are now detected and displayed, including AWS profiles, container environment, `kubectl` context, Node.js, Ruby, and Terraform workspace.

Some of these sections are disabled by default, so see their respective config options for more details and how to enable them:

- *LP_ENABLE_AWS_PROFILE*
- *LP_ENABLE_CONTAINER*
- *LP_ENABLE_KUBECONTEXT*

- *LP_ENABLE_NODE_VENV*
- *LP_ENABLE_RUBY_VENV*
- *LP_ENABLE_TERRAFORM*

7.1.3 Linux sysfs interface

Both battery and temperature information now support reading from the sysfs interface on Linux. This means `sensors` and `acpi` are no longer required to get battery and temperature information, and the data source is much faster.

7.1.4 Wireless Signal Strength

A new feature exists to track wireless signal strength, and display the value if it falls below a threshold, similar to the battery feature. See *LP_ENABLE_WIFI_STRENGTH* for more information.

7.1.5 Shell Nesting Level

A new feature to display the number of nested shells. See *LP_ENABLE_SHLVL* for more information.

7.1.6 ShellCheck

The whole project is now checked with ShellCheck, which has already caught a few issues, and will help prevent regressions.

7.1.7 Bash-preexec Compatibility

Liquid Prompt now supports running along side bash-preexec. Simply load Liquid Prompt *after* bash-preexec.

7.2 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.

7.2.1 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.

7.2.2 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*.

7.2.3 Example Themes

Liquid Prompt 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*.

7.2.4 Data Sources

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

7.2.5 Path Advanced Formatting

The current directory path has had an overhaul, now supporting formatting for different path sections, highlighting the last directory and the VCS repository root directory. Any shortened directories are lowlighted instead. Multiple path shortening methods are now supported as well.

See *LP_PATH_METHOD* and *LP_COLOR_PATH* for more information.

7.2.6 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*.

7.2.7 Version Control Tracking updates without directory change

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

7.2.8 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!

7.2.9 Documentation

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

7.2.10 Unit Tests

A whole suite of unit tests has been added to test the data and utility functions. A handful of bugs were caught using the tests, so the investment has already paid off!

7.2.11 Directory Stack Feature

Shell directory stack display is now in the prompt! If there are directories on the stack (from `pushd`), the number of directories in the stack is shown next to the current path. See *LP_ENABLE_DIRSTACK* for more information.

7.3 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.

7.3.1 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_ENABLE_RUNTIME_BELL* and *LP_RUNTIME_BELL_THRESHOLD* config options.

7.3.2 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*.

7.3.3 Preset Color Aliases

The `5` value of the basic colors is often named "magenta", but in Liquid Prompt 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`.

7.3.4 Speed Improvements

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

CHAPTER

EIGHT

UPGRADING LIQUID PROMPT

We try our best to make new versions of Liquid Prompt 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.

8.1 Version 2.1 Upgrade Notes

Upgrading to version 2.1 is almost completely painless, as there is only one deprecation.

8.1.1 Deprecations

Public Deprecations

$LP_ENABLE_FQDN

Replaced by *LP_HOSTNAME_METHOD*.

Replace a set statement like:

```
LP_ENABLE_FQDN=1
```

with:

```
LP_HOSTNAME_METHOD=full
```

Private Deprecations

_lp_as_text()

Replaced by *__lp_strip_escapes()*.

Replace a statement like:

```
text="$(_lp_as_text "$string")"
```

with:

```
local ret
__lp_strip_escapes "$string"
text="$ret"
```

8.2 Version 2.0 Upgrade Notes

Most of the changes in 2.0 are in private functions and variables. There are a few public API changes that could impact users: *$lp_err*, *$LP_DISABLED_VCS_PATH*, *$LP_PATH_DEFAULT*, *$PROMPT_DIRTRIM*, *$LP_PATH_KEEP=-1*, and *_lp_title()*. The rest are private API changes, but are still documented here.

8.2.1 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_format* 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()

Removed and replaced by *_lp_path_format*. Support for `\w` in PS1 has been dropped.

_lp_set_prompt()

Renamed to *__lp_set_prompt*.

Recommended to not use this internal function.

_lp_shorten_path()

Removed and replaced by *_lp_path_format()*.

Replace assignment statements like:

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

with:

```
local lp_path_format
_lp_path_format "$LP_COLOR_PATH" "$LP_COLOR_PATH_LAST_DIR" "$LP_COLOR_PATH_VCS_ROOT" "
↪$LP_COLOR_PATH_SHORTENED" "/" "$LP_COLOR_PATH_SEPARATOR"
cwd=$lp_path_format
```

_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
```

8.2.2 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_PATH_DEFAULT

Replaced by *LP_PATH_METHOD*.

If one of the many new shortening methods does not effectively replace your use case, please open an enhancement request.

$PROMPT_DIRTRIM

`$PROMPT_DIRTRIM` calculation is no longer supported. Replaced by *LP_PATH_METHOD* set to `truncate_chars_from_path_left`.

$LP_PATH_KEEP=-1

LP_PATH_KEEP set to `-1` is replaced by *LP_PATH_METHOD* set to `truncate_to_last_dir`.

_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

NINE

INDICES AND TABLES

- genindex
- search

INDEX