Introduction to Powerline the statusline plugin for VIM

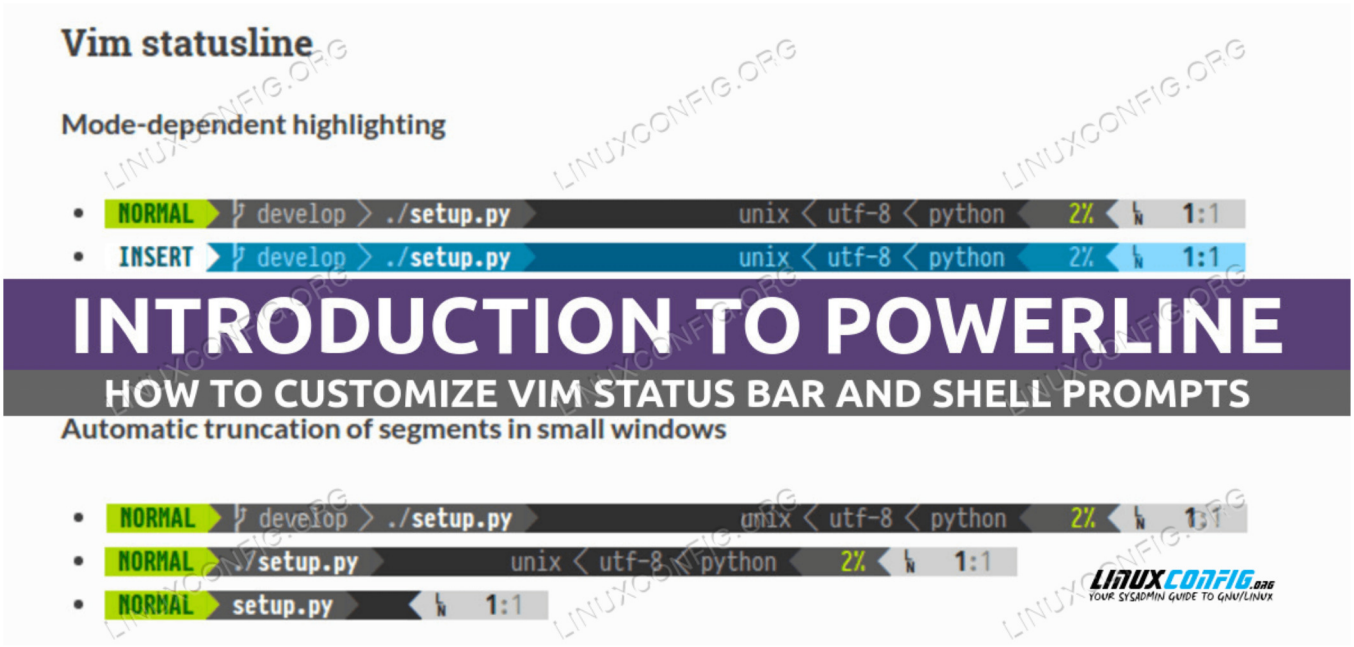
7 February 2022 by Egidio Docile

Vim is one of most used and famous text editors on Linux and other Unix-based operating systems. It is free and open source software, based on the original Vi editor (Vim stands for Vi IMproved) and mainly developed by Bram Moolenaar. The editor can be setup by editing the `~/.vimrc` configuration file, and by using a vast range of plugins. In this article we see how to improve and customize the Vim status bar by installing Powerline, a nice plugin written in Python which can be also used to customize the Bash and Zsh shell prompts.

In this tutorial you will learn:

- How to install Powerline and powerline-fonts
- How to check if Vim was compiled with Python support
- How to use Powerline to customize Vim status bar
- How to use Powerline to customize Bash prompts

- How to use Powerline to customize zsh prompts



Introduction to Powerline the statusline plugin for VIM

Software requirements and conventions used

Software Requirements and Linux Command Line Conventions

Category	Requirements, Conventions or Software Version Used
System	Distribution-independent
Software	Vim, Python3, Powerline
Other	None
Conventions	# – requires given linux-commands to be executed with root privileges either directly as a root user or by use of `sudo` command $ – requires given linux-commands to be executed as a regular non-privileged user

Installing Powerline and powerline-fonts

Powerline is an open source software written in Python; we have basically two ways to install it: we can either use `pip`, the Python package manager, or our distribution native package manager.

Installing Powerline using pip

The cross-distribution way to install Powerline is by using `pip`, the Python package manager. Using this installation method has the advantage of providing always the latest stable version of the package. The `pip` command should always be launched as an unprivileged user in order to perform a user-level installation. To install Powerline, we need to run:

```
$ pip install --user powerline-status
```

When performing per-user installation with the `--user` option, libraries and binary files on Linux are installed under the `~/.local` directory by default.

Powerline fonts are fonts which contains some glyphs that are used as part of the status bar and prompts created by Powerline. When Powerline is installed via our favorite distribution package manager, the powerline-fonts are automatically included as a dependency; if we perform an installation using `pip`, however, we must install them manually. All we have to do is to clone the dedicated git repository:

```
$ git clone https://github.com/powerline/fonts
```

Once the repository has been cloned on our machine, we can launch the `install.sh` script contained in it to install the fonts. Since we are performing the installation as an unprivileged user, the fonts will be installed under the `~/.local/share/fonts` directory. The fc-cache command will be launched automatically by the script, to refresh the fonts cache.

Installing Powerline with a distribution package manager

Since Powerline is available in the official repositories of some of the most used Linux distributions, as an alternative, we can install it by using our favorite package manager. On Fedora, for example, we should use the following command:

```
$ sudo dnf install powerline
```

On Debian, and Debian-based distributions, instead, we can run:

```
$ sudo apt install python3-powerline
```

Powerline is available in the Archlinux "Community" repository, so we can install it via the `pacman` package manager. On Arch the powerline fonts are not include a dependency, therefore they must be installed explicitly:

```
$ sudo pacman -Sy powerline powerline-fonts
```

Using Powerline to customize the Vim status line

To be able to use Powerline with Vim we must make sure the editor was compiled with Python support. To verify this we can run the following command:

```
$ vim --version
```

The command displays a series of information about the Vim version installed and the features it was compiled with. If a feature is preceded by a A `+` sign it means that Vim has been compiled with support for it, so we should check the status of the `python` or `python3` flags. A shortcut to verify Vim was compiled with support for Python, is to use the following command:

```
$ vim --version | grep -i +python
```

If the command returns no results, then Vim has no support for Python, and, to use Powerline, we should re-compile it with the appropriate flags (notice that some distributions like Debian provide different packages for Vim. The standard `vim` package provides a version of the editor which is compiled without support for Python. A more featured version with Python support is included in the `vim-nox` package), otherwise we are good to go!

Editing vim configuration file

To start using Powerline with Vim, all we have to do is to add some lines in our `.vimrc` configuration file. In this example I suppose support for Python3 exists; if using Python2, just change the interpreter name accordingly:

```
 python from powerline.vim import setup as powerline_setup
python powerline_setup()
python del powerline_setup
```

Once the above content is written in the `~/.vimrc` file, to make the changes effective we can either close and re-open Vim, or just re-source the configuration file by entering the editor command mode (`:`) and launching the following command:

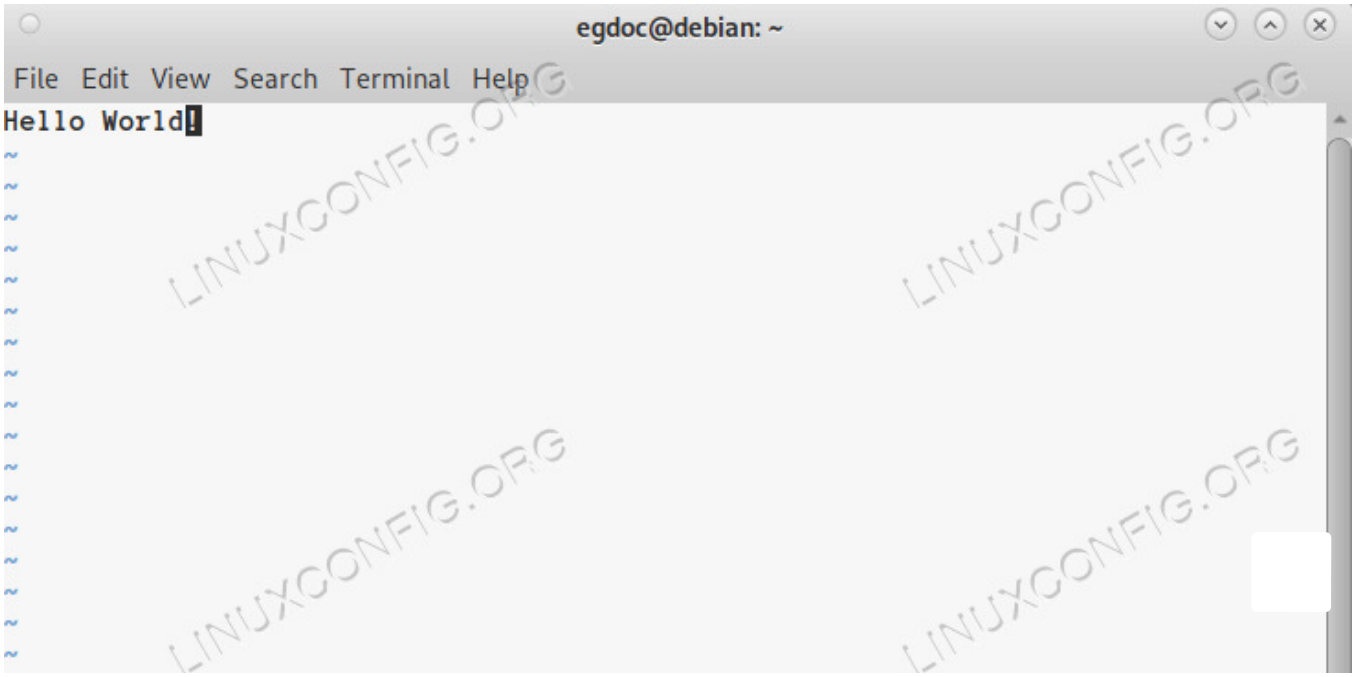
```
:so ~/.vimrc
```

If we open Vim we and load the new configuration, at this point we probably don't see anything new, why? This behavior is expected since by default the status bar is displayed only if at least **two** windows exist. To make the status bar always be displayed we must change the value of the `laststatus` option. This option takes three possible values:

- 0: The status bar is never displayed
- 1: This is the default: the status bar is displayed only if there are at least two windows
- 2: The status bar is always displayed

We want to set `2` as value, therefore inside the configuration file we add the following line:

```
set laststatus=2
```



Vim Powerline status bar

Once the new changes are loaded, the Powerline status bar should be correctly displayed inside vim:

If the status bar is displayed correctly but there are missing symbols, and you installed powerline-fonts manually, make sure you selected the patched font variants (those with the "for powerline" suffix), as the terminal emulator fonts, if using command line vim. If using `gvim` instead, you can set the font by using the `guifont` option.

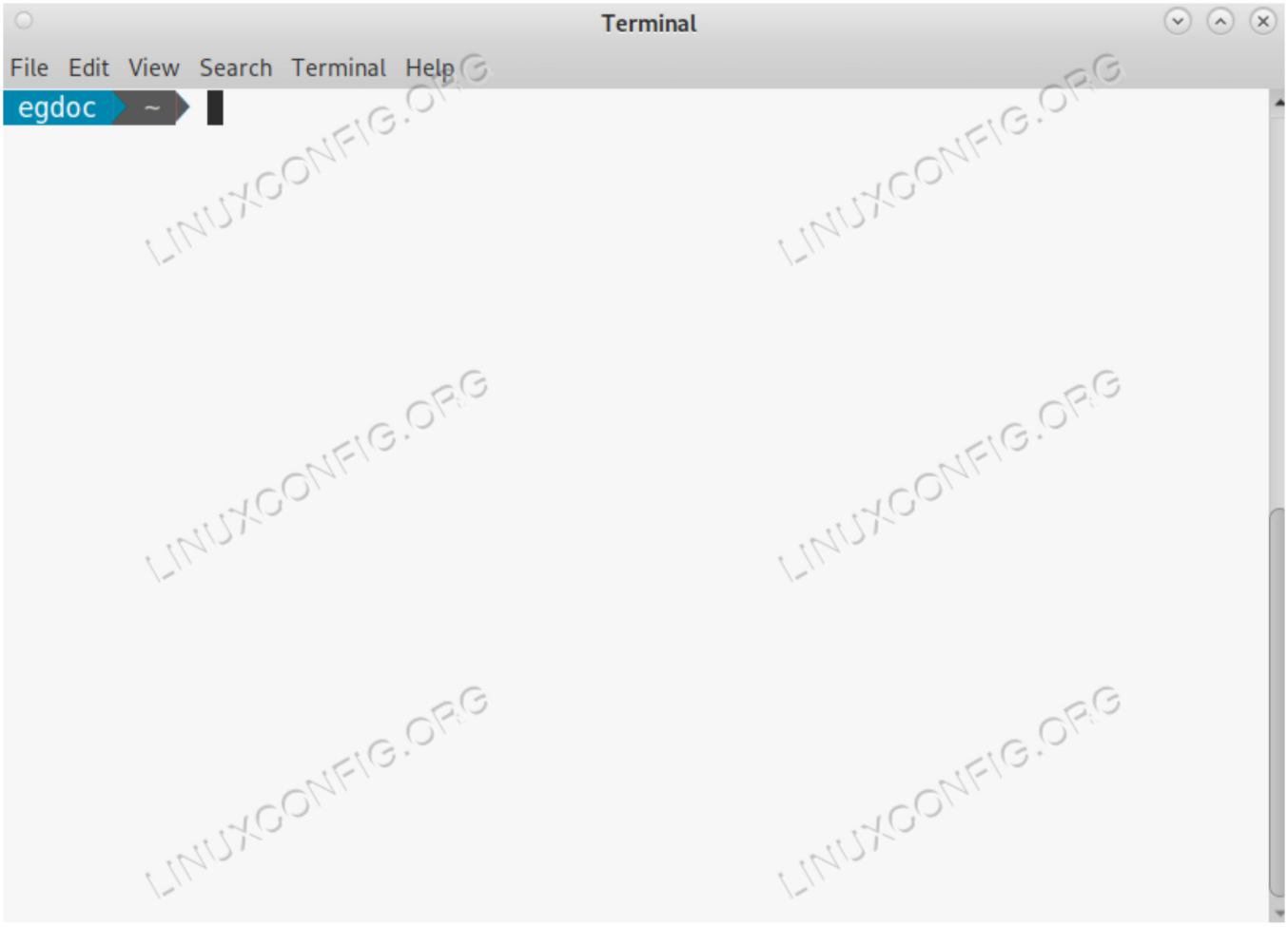
Using Powerline to customize shell prompts

The Powerline status bar can be used in other applications, not only in Vim. For example, we can use it to add some bell and whistles to our shell prompts. To use Powerline with Bash or Zsh, we must first know the Powerline installation path: this depends on the method we used to install it. If we installed the plugin with `pip` and Python3, the path should be `~/.local/lib/python<version>/site-packages/powerline`, where `<version>` should be substituted with the version of Python actually used for the installation. For example, if we are using Python 3.9, the path should be: `~/.local/lib/python3.9/site-packages/powerline`. If we installed Powerline using our distribution native package manager, instead, the path varies depending on the distribution we are using. In Debian and Archlinux, the Powerline installation path would be: `/usr/share/powerline`; on Fedora, instead, the path is: `/usr/share/powerline`.

Customizing Bash prompt

To customize Bash prompts with Powerline, all we have to do is to add the following content to the `~/.bashrc` file (if Powerline was installed by using Fedora native package manager, the path of the script to be sourced should be changed to: `/usr/share/powerline/bash/powerline.sh`):

```
powerline-daemon -q
POWERLINE_BASH_CONTINUATION=1
POWERLINE_BASH_SELECT=1
source /bindings/shell/powerline.sh
```



The Bash Powerline prompt

The `powerline-daemon -q` command starts the `powerline-daemon`, this is nee

to achieve better performances when opening the terminal. Here is a screenshot of the Bash Powerline prompt:

Using Powerline prompt with Zsh

In a previous article we talked about Zsh. Z-shell is a modern shell which can be used as an alternative to Bash; Powerline can be used also with it. If we want to use Powerline with Zsh, all we need to do is to **source** the `powerline.zsh` script from the `~/.zshrc` configuration file:

```
source /bindings/zsh/powerline.zsh
```

Once again, if we installed Powerline via `dnf` on Fedora, the path of the script is slightly different: `/usr/share/powerline/zsh/powerline.zsh` .

Summary

In this article we saw how to install and configure one very nice utility we can use to customize the statusbar/prompt of many applications: Powerline. We saw how to use it to customize the Vim status bar and the prompts of the Bash and Zsh shells. Powerline supports also other shells (like Fish and Rcsh) and status widgets like the i3-bar, which is the i3 window manager default status bar. For instructions about how to use Powerline with those applications please take a look at the official documentation.

Related Linux Tutorials:

- Zsh shell installation and configuration on Linux
- Things to install on Ubuntu 20.04
- Vim editor basics in Linux
- List of LaTeX editors and compilers on Ubuntu 20.04...
- Linux Vs. Unix: What's the Difference?
- How to work with the Woocommerce REST API with Python
- Things to do after installing Ubuntu 20.04 Focal Fossa Linux
- How to build a Tkinter application using an object...

- Python Regular Expressions with Examples
- How to perform HTTP requests with python - Part 1:...

System Administration

administration, commands, filesystem, installation

How to build a Tkinter application using an object oriented approach

Ubuntu mirrors

Comments and Discussions

LINUXCONFIG.ORG FORUM

Start Discussion 0 replies

NEWSLETTER

Subscribe to Linux Career Newsletter to receive latest news, jobs, career advice and featured configuration tutorials.