Hyper-Prompt

Release 1.1.0

Aug 15, 2020

Contents

A Highly Customize-able prompt for your shell

CHAPTER 1

Contents

1.1 Installation

1.1.1 Requirements

- Python3
- Powerline Fonts : https://github.com/powerline/fonts

1.1.2 Getting Started

- from pip:

```
pip install hyper-prompt
```

- from github:

```
git clone https://github.com/artbycrunk/hyper-prompt
cd hyper-prompt
python setup.py install
```

1.2 Configuration

1.2.1 Configuration File

Hyper prompt will lookup multiple locations for a config file. The config file provides options on how to display the prompt.

A valid config will be looked up in the following order.

- $PWD/hyper_prompt.json

- $HOME/.hyper_prompt.json
- $HOME/.config/hyper_prompt/config.json

Note: If no config file is available the fallback is a hardcoded default list of segments.

Example config:

```
"theme": "default",
"mode":"patched",
"segments": [
    "username",
    {
        "type": "virtual",
        // a user built segment which is discoverable via the python path
        "module": "hyper_prompt.segments.virtual"
    }
]
```

1.2.2 Shell Configuration

Bash

Add the following to your *.bashrc* file:

```
function _update_ps1() {
    PS1=$(hyper-prompt $?)
}

if [[ $TERM != linux && ! $PROMPT_COMMAND =~ _update_ps1 ]]; then
    PROMPT_COMMAND="_update_ps1; $PROMPT_COMMAND"
fi
```

Zsh

Add the following to your *.zshrc*:

```
function prompt_precmd() {
    PS1="$(hyper-prompt --shell zsh $?)"
}

function add_prompt_precmd() {
  for s in "${precmd_fn[@]}"; do
    if [ "$s" = "prompt_precmd" ]; then
      return
    fi
  done
  precmd_fn+=(prompt_precmd)
}

if [ "$TERM" != "linux" ]; then
    add_prompt_precmd
fi
```

Fish

Add the following to your *~/.config/fish/config.fish*:

```
function fish_prompt
    hyper-prompt --shell bare $status
end
```

Tcsh

Add the following to your *.tcshrc*:

```
alias precmd 'set prompt="`hyper-prompt --shell tcsh $?`"'
```

1.3 Segments

Segments are the building blocks of hyper-prompt

You can mix and match different segments to build your prompt. They can also depends on each other is certain cases.

1.3.1 Segment Configuration

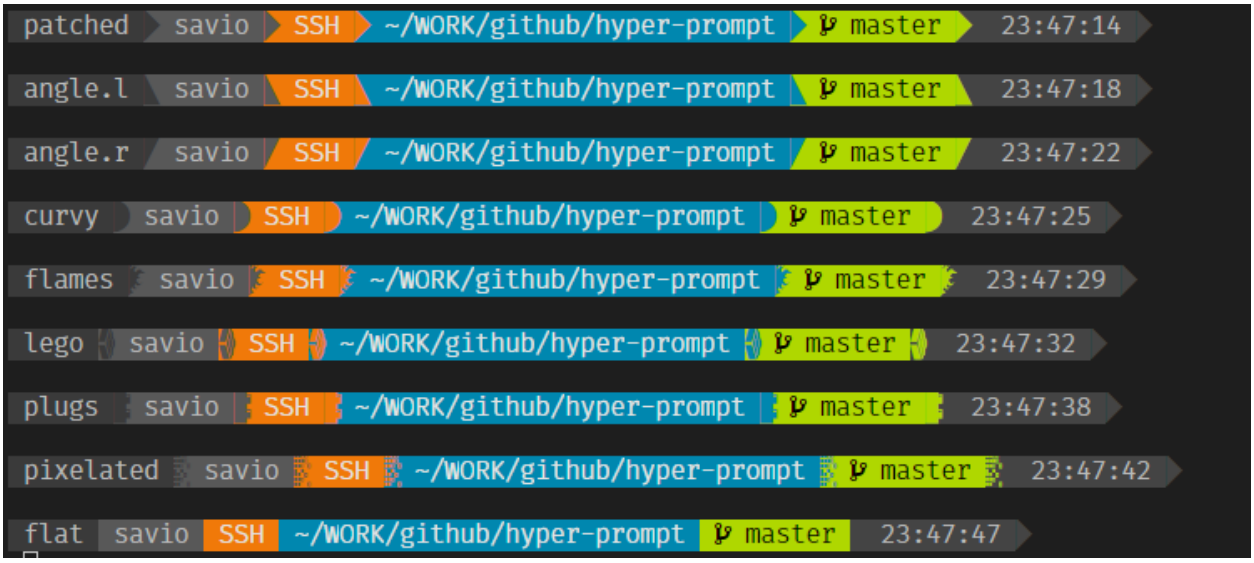
Each segment can have its only configurable attribute, if you want to configure a segment beyond its default you can do some like the following example.:

```
{
  "segments": [
      "ssh",
      {
          "type": "cwd",
          "show_readonly": true
      },
      {
          "type":"time",
          "show_symbols":false,
          "separator":"patched"
      }
  ]
}
```

1.3.2 Separators & Symbols

Each segment is seperated by a default symbol from the powerline font, this can be changed by the *separator* option in the config.

Available separators options are: patched, angle.l, angle.r, curvy, flames, lego, plugs, pixelated, flat Refer to the following image of the style of each separator.



Each segment also has the option of displaying symbols, to better represent the segment info, this can be globlally toggled using the *show_symbols* option in the config.

1.3.3 All Segments

cwd

The options for the **cwd** segment are:

- **mode**: If *plain*, then simple text will be used to show the cwd. If *dironly*, only the current directory will be shown. Otherwise expands the cwd into individual directories.
- **max_depth**: Maximum number of directories to show in path.
- **max_dir_size**: Maximum number of characters displayed for each directory in the path.
- **full_cwd**: If true, the last directory will not be shortened when *max_dir_size* is used.

env

var skip_undefined

git

hostname

newline

root

ssh

time

username

virtualenv

1.4 Themes

CHAPTER 2

Indices and tables

- genindex
- search