suckless-utils

Release 6.4

Alexis Jhon Gaspar

Nov 01, 2023

GETTING STARTED

`suckless-utils` is a collection of applications that aims to form a desktop environment that is modular, flexible and lightweight (for the RAM and storage).

- **Modular** - it aims for various functionality that can be turned off or on in the source code level, which could make for various personal versions of each component.
- **Flexible** - it aims to be flexible, with the ability to have various functionality, window management layouts and down-to-source code customization aiding for that, as well as the ease of hacking due to the `-flexipatch` forks by bakkeby implemented preprocessor directives.
- **Lightweight** - compared to other desktop environments, the suite clocks in at 700+ MB in RAM usage when idle (but actual mileage may vary), and it's small source code size (only clocking in at 18.2MB not including `.git`) makes it the lightest desktop environment that you could compile yourself.

It features things such as EWMH support, IPC and `fsignal`, `pywal` color scheming, **dynamic** window management (tiled, floating, tabbed and stacked window types), desktop icons (`nemo-desktop`), notfications (`dunst`), and more!

Check out the sidebar to get started.

Note: This project is under heavy development.

CHAPTER ONE

CONTENTS

1.1 Installing `suckless-utils`

Tip: All of the steps should be reproducible within any modern Linux distros with internet access, but with minor tweaking.

Note: Some components might be also be available on your distro's repository (for example the Arch User Repository). Refer to your distro's package management for more info.

1.1.1 Dependencies

Before installing all components, we need the dependencies:

- `git`
- `xorg` (including drivers of course)
- `base-devel` (or `build-essential/s`)
- `libX11` (`-devel` or `-dev`)
- `libXft` (`-devel` or `-dev`)
- `libXcb` (`-devel` or `-dev`)
- `libXrender` (`-devel` or `-dev`)
- `libXinerama` (`-devel` or `-dev`)
- `freetype` (`-devel` or `-dev`)
- `fontconfig` (`-devel` or `-dev`)
- Nerd Fonts (Hack as default, can be changed manually)
- `imlib2` (`-devel` or `-dev`)
- `picom` (for transparency)
- `feh` (optional, if one decided not to use `pywal`)
- `pywal` (for colors/wallpaper)
- `slop` (for the `riodraw` patch in `dwm`)

- `yajl` (for handling IPC in `dwm` with `dwm-ipc` enabled)
- `eww` (optional, but recommended)
- `jgmenu` (for handling context menus, recommended but `xmenu` could be used as an alternative.)
- `libexif` (`-devel` or `-dev`) (for `nsxiv`)
- `jq` (for handling eww notifications, as well for the `adelle-theme` script.)
- `pamixer` (for the media related scripts.)
- `nemo` (for handling desktop icons, though other file managers with similar functionality could be used.)

In addition, if you use the Termux application for Android, you must **also** install these:

- `termux-X11` repo (via main Termux app)
- `proot/chroot`
- PulseAudio (if you like audio support)
- Display client of choice:
  - TigerVNC
  - VNC client

  or

  - a XSDL client

  or

  - Termux:X11 (both the `apk` and the companion `deb` package.)

For installing `spmenu`:

Tip: You probably don't need the Wayland libraries as well, you could build `spmenu` with only X11 by running `meson setup build -Dwayland-=false`.

- `wayland-client` (`-devel` or `-dev`, for Wayland support)
- `wayland-scanner` (`-devel` or `-dev`, for Wayland support)
- `wayland-protocols` (`-devel` or `-dev`, for Wayland support)
- `xbcommon` (`-devel` or `-dev`, for Wayland support)
- `pango` (`-devel` or `-dev`)
- `cairo` (`-devel` or `-dev`)
- `libconfig` (`-devel` or `-dev`)
- OpenSSL or `libssl` (`-devel` or `-dev`)
- `meson`

To make the tabbed windows functionality to work:

Note: This uses `tabbed` instead of it being integrated in `dwm` itself due to modularity reasons, as well as ease of implementation.

- `cut`

- `xargs`
- `grep`
- `pstree`
- `sed`
- `wmctrl`
- `xdotool`
- `xprop`
- `xwininfo`

Note: Refer to `patches.def.h` and `config.mk` for additional dependencies, as well as tweaks for other Unix-based operating systems such as OpenBSD.

1.1.2 Setting up build environment and compilation

1. Install the dependencies defined above.
2. Clone the repository (`git clone --recurse-submodules`)
3. Change directory to the `suckless-utils` directory.
4. Before compilation, remove `config.h` and `patches.h` (if exists) to ensure loading of defaults.
5. Start building `dwm` first by using `make clean` then `make` then `sudo make install` (For `nsxiv`, it's `make install-all`.)
6. After that, build anything except `spmenu` and `slim`.

Note: It is important to build anything except `spmenu` and `slim` first as it uses a different build system than most of the components.

7. Setup `spmenu` by running `meson setup build`. Pass `-Dwayland-=false` for disabling Wayland support.
8. Run `ninja -C build` for building spmenu binaries.
9. Install `spmenu` by running `meson install -C build`. It would prompt for root access if necessary.

Tip: Installing `slim` is optional as well, if one already have SDDM or GDM installed.

10. Make a build folder inside the `slim` directory.
11. Generate the `Makefile` via `cmake`. Make sure the `PREFIX` variable is set on the `/usr` directory.
12. Then run `make` and `sudo make install` as usual.
13. Set up the systemd service as well. If not, tweaking is necessary.
14. And it's built!

1.1.3 Installing scripts

The suite also utilize a lot of scripts as well. For example, `layoutmenu` handles setting layouts easily in `dwm`.

Installing most scripts are just as easy as copying and pasting it to `$PATH` and changing permissions if neccessary.

Note: It is recommended to have `$HOME/.local/bin` in the `$PATH` variable to avoid conflicts.

Installing some scripts however might need some effort to install. For example, `adelle-theme` needs `wget`, `tabb` needs the same dependencies as enabling tabbed window functionality, and the `quoter` script needs `messages.txt` to be placed on $HOME.

Some are reliant to `jgmenu`, notably `shutdown` and `layoutmenu`.

In `spmenu` some actually needs to be compiled manually. Here's the dependencies for some:

`clipmenu-spmenu` dependencies:

- `xsel`
- `clipnotify`

`screenshot-spmenu` dependencies:

- `curl`
- `xclip` (X11)
- `maim` (X11)
- `wayshot` (Wayland)
- `wl-clipboard` (Wayland)
- `slurp` (Wayland)

`wallpaper-spmenu` dependencies:

- `xwallpaper`

Scripts are also necessary to make `dwmblocks-async` working, as well as the complete `eww` functionality.

Congratulations! You just installed the suite! But hold your horses, we need some more set up, especially with the configuration files.

1.2 First Setup

Congrats! You just installed the suite! But you must need to do these before **actually** starting them. Luckily it's just for the first time you run this. No need for further pain after these are done.

1.2.1 Initializing `pywal`

First copy all of the things in `config` to `$HOME` and rename it to `.config`.

To all of the colorschemes to work, hard link the generated colorscheme for everything in `$HOME/.cache/wal/` to each of the components:

- `cava-config` hard link to `$HOME/.config/cava/config`
- `colors.scss` hard link to `$HOME/.config/eww/colors.scss`
- `dunstrc` hard link to `$HOME/.config/dunst/dunstrc`
- `jgmenurc` hard link to `$HOME/.config/jgmenu/jgmenurc`
- `vis` hard link to `$HOME/.config/vis/colors/pywal`

and you're basically done, except if you use SLiM.

SLiM is a bit different than the others, necessitating `sudo`. To actually reload, run the `slim-reload` in the same folder. This would prompt `sudo` as you're actually modifying the theme in `/usr` to replace the colors aand wallpaper.

and you're finally done with `pywal`!

1.2.2 Configuring some scripts

To configure the output for the `quoter` script, you need to edit the `messages.txt` file in the home directory.

Next, you need to modify **both** `$HOME/.config/eww/eww.yuck` and `$PATH/sb-forecast` if neccessary, to change location.

Lastly, you need to modify `$HOME/.config/eww/eww.yuck` and `$HOME/.config/eww/eww.scss` to adjust width settings and variables.

And that's it! Hope you enjoy the desktop!

1.3 Components

In this section, we are documenting basic functionality and information about each and every `suckless-utils` component.

Note: As stated in the installation part, Some components might be also be available on your distro's repository (for example the Arch User Repository). Refer to your distro's package management for more info.

Tip: You could change the keybinds of most components, but I don't usually reccommend doing it as it might conflict with the other component's keybinds.

1.3.1 Core components