RDE

Tools for managing reproducible development environments

Nicolas Graves

February 2, 2025

Outline

My linux journey

RDE

How does RDE build upon GNU Guix ?

RDE's future

More info (not actually presented, for questions and transcript)

Windows –> Ubuntu-based (2013-14)

- Removing proprietary OS is possible!
- FOSS is great!
- Minimalism and effiency is great!
- (l|x)ubuntu revives old computers!

Ubuntu –> Archlinux (2020)

- UNIX Philosophy is great!
- Crafting your system is fun!
- but it can be tedious. . .

Archlinux –> RDE / GNU Guix (2021)

UNIX Philosophy is great. . .	. . . but forgets transaction costs.
Crafting your system is fun. . .	. . . but life and loss happens!
but it can be tedious. . .	. . . but it doesn’t have to be!

Archlinux –> RDE / GNU Guix (2021)

. . . but forgets transaction costs.	⇒ GNU Guile
. . . but life and loss happens!	⇒ Guix Reproducibility
. . . but it doesn't have to be!	⇒ RDE framework

RDE

Developer and power user friendly GNU/Linux distribution building upon Guix and its tools (Guile API, guix daemon, guix system, guix home).

Figure: Guix FOSDEM 2023 presentation

What is RDE to Linux?

GNU/Linux	Emacs	Emacs
RDE	Spacemacs	Doom Emacs
Features	Layers	Modules
GNU Guix	use-package	straight.el

Features are blocks of configuration that provide certain functionality for a user, such as setting up your email, adding your GnuPG keys, or configuring your window manager.

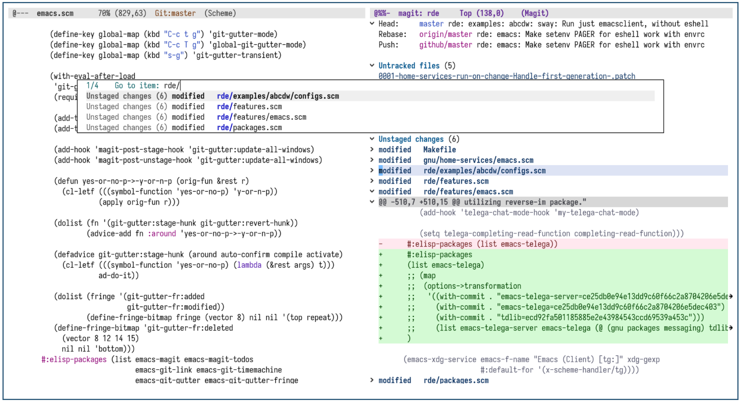
Minimalist and sane

Ergonomic	Sane keybindings, good contrast, readable fonts.
Lightweight	battery efficient, wayland, fast native apps.
Offline	most of workflows/apps should work without network.
Attention	minimal use of notifications.
Standards	comply with the XDG Base Dirs Specification.

Tools

Purpose	Tool
Window Manager	Sway
Terminal	Alacritty/Foot
Login Shell	GNU Bash
Interactive Unix Shell	Zsh
Service Manager/Init System	GNU Shepherd
Package Manager	GNU Guix
Filesystem	Btrfs
Multimedia Framework	PipeWire
Video Player	mpv
Everything Else ;)	GNU Emacs

How does it look?



```
@---  emacs.scm      70% (829,63)  Git:master  (Scheme)

        (define-key global-map (kbd "C-c t g") 'git-gutter-mode)
        (define-key global-map (kbd "C-c T g") 'global-git-gutter-mode)
        (define-key global-map (kbd "s-g") 'git-gutter-transient)

        (with-eval-after-load
         'git-g
         (requi

         (add-t
         (add-t

         (add-hook 'magit-post-stage-hook 'git-gutter:update-all-windows)
         (add-hook 'magit-post-unstage-hook 'git-gutter:update-all-windows)

         (defun yes-or-no-p->-y-or-n-p (orig-fun &rest r)
           (cl-letf (((symbol-function 'yes-or-no-p) 'y-or-n-p))
                    (apply orig-fun r)))

         (dolist (fn '(git-gutter:stage-hunk git-gutter:revert-hunk))
                 (advice-add fn :around 'yes-or-no-p->-y-or-n-p))

         (defadvice git-gutter:stage-hunk (around auto-confirm compile activate)
           (cl-letf (((symbol-function 'yes-or-no-p) (lambda (&rest args) t)))
                    ad-do-it))

         (dolist (fringe '(git-gutter-fr:added
                           git-gutter-fr:modified))
                 (define-fringe-bitmap fringe (vector 8) nil nil '(top repeat)))
         (define-fringe-bitmap 'git-gutter-fr:deleted
           (vector 8 12 14 15)
           nil nil 'bottom)))
      #:elisp-packages (list emacs-magit emacs-magit-todos
                             emacs-git-link emacs-git-timemachine
                             emacs-git-gutter emacs-git-gutter-fringe
```

```
1/4    Go to item: rde/
Unstaged changes (6) modified   rde/examples/abcdw/configs.scm
Unstaged changes (6) modified   rde/features.scm
Unstaged changes (6) modified   rde/features/emacs.scm
Unstaged changes (6) modified   rde/packages.scm
```

```
@%%-  magit: rde     Top (138,0)    (Magit)
⌄ Head:     master rde: examples: abcdw: sway: Run just emacsclient, without eshell
  Rebase:   origin/master rde: emacs: Make setenv PAGER for eshell work with envrc
  Push:     github/master rde: emacs: Make setenv PAGER for eshell work with envrc

⌄ Untracked files (5)
  0001-home-services-run-on-change-Handle-first-generation-.patch

⌄ Unstaged changes (6)
> modified   Makefile
> modified   gnu/home-services/emacs.scm
> modified   rde/examples/abcdw/configs.scm
> modified   rde/features.scm
⌄ modified   rde/features/emacs.scm
⌄ @@ -510,7 +510,15 @@ utilizing reverse-im package."
            (add-hook 'telega-chat-mode-hook 'my-telega-chat-mode)

            (setq telega-completing-read-function completing-read-function)))
-      #:elisp-packages (list emacs-telega))
+      #:elisp-packages
+      (list emacs-telega)
+      ;; (map
+      ;;  (options->transformation
+      ;;   '((with-commit . "emacs-telega-server=ce25db0e94e13dd9c60f66c2a8704206e5de
+      ;;     (with-commit . "emacs-telega=ce25db0e94e13dd9c60f66c2a8704206e5dec403")
+      ;;     (with-commit . "tdlib=ecd92fa501185885e2e43984543ccd69539a453c")))
+      ;;     (list emacs-telega-server emacs-telega (@ (gnu packages messaging) tdlib
+      )

        (emacs-xdg-service emacs-f-name "Emacs (Client) [tg:]" xdg-gexp
                           #:default-for '(x-scheme-handler/tg))))
> modified   rde/packages.scm
```

Reproducible and stateless

Building on GNU Guix:

- any setup/configuration can be easily replicated
- roll-back if you break any configuration

⇒ A huge garanty for further system crafting!
We try to make all state (directories) explicit and syncable or temporary. ⇒ Help yourself avoid loosing data!

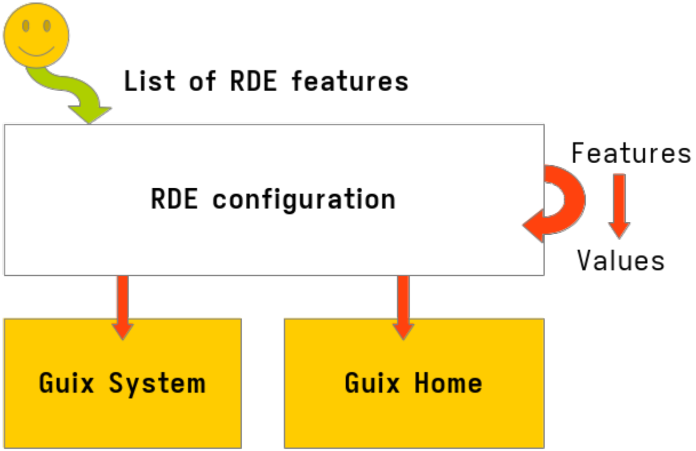
Hackable

- GNU Guile + Guix APIs
- No compromission on extensibility : Easy to throw out or modify any part of setup.

RDE is opiniated, but is composable and flexible enough:

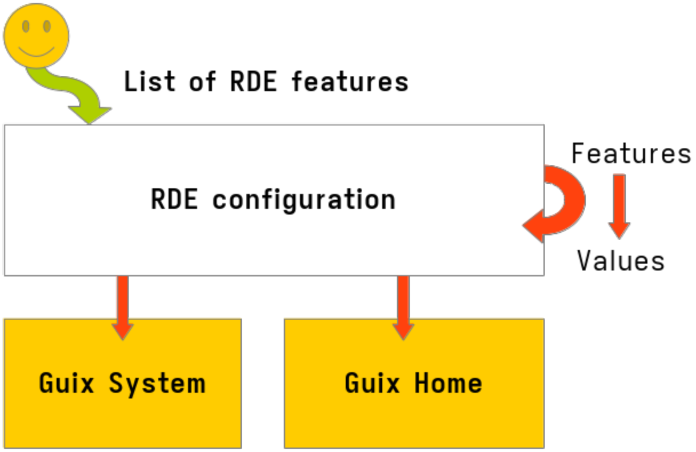
- X language development,
- science,
- sysadmin/devops...

RDE config



Friends with GNU Guix, compatible with GNU Guix development

Features



A feature unifies Home/System configuration to make it easier for the user to opt-in/out of a functionality.

A non-exhaustive list of features

email	feature-mail-settings, -msmtp, -notmuch, -l2md
wm	feature-sway, -waybar, -swayidle, -swaylock, -sway-screenshot
programming	feature-guile, -python, -prolog, -go, -ocaml, -lisp
emacs (general)	feature-emacs, -emacs-completion, -emacs-vertico, -emacs-help, -emacs-eat
emacs (dev)	feature-emacs-git, -emacs-eglot, -emacs-dape
emacs (org)	feature-emacs-org, -emacs-org, -emacs-org-citation, -emacs-org-roam

Value added above Guix: cross-configuration

What if I want a piece of config in Software A if and only if Software B is present ?
$\Rightarrow$ Simple! Conditionnally check the presence of software B, and include your piece of config only in this case.

Value added above Guix: Serializer debate

What if I want a service that is not yet in Guix or RDE ?
⇒ We have serializers to do the job quickly, already :

- css
- elisp
- ini
- json
- lisp
- nginx
- utils

⇒ Write guix home/system services in a more flexible way.

RDE's future

RDE is quite mature and has been usable for years!
Work in progress:

- Programming languages / AI integration
- Statelessness / Impermanence
- New API to allow users to define an <rde-config> based on options rather than features
- Generalizing package propagation
- and much more!

$\Rightarrow$ Good dynamic (along with the Guile ecosystem!)

Support the project

Principles

The principles I presented earlier are organized as such:

Main

Ergonomic	Sane keybindings, good contrast, readable fonts.
Reproducible	Setup can be easily replicated.
Hackable	Easy to throw out or modify any part of setup.
Stateless	All state must be explicit and syncable or temporary.

Secondary

Lightweight and battery efficient	wayland, fast native apps.
Offline	most of workflows and apps should work without network.
Attention-friendly	minimal use of notification and other distractive things.

I want to try RDE's ideas, but I use NixOS !

Guix is great too ;p Or you can try ordenada, a port of RDE for NixOS