(emacs)

How to start emacs with a custom user-emacs-directory

56

21

I'm working on a custom and small Emacs configuration that I want to share with some friends as a git repository for them to use as a baseline for their own future configurations.

For this I need some way to test my configuration and the simplest solution I can come up with is something like:

```
$ emacs --eval "(setq user-emacs-directory \"~/Code/my_custom_emacs.d/\")"
```

But I can't seem to make it work.

Any help much appreciated.

init-file

edited Dec 10 '14 at 7:13

asked Dec 10 '14 at 6:58

Mattias Bengtsson
619 1 6 15

3 I read over Appendix C Command Line Arguments for Emacs Invocation but I did not find a simple command-line option to start Emacs with a custom `.emacs.d` directory unless you change HOME, which seems problematic to me. People have provided workarounds below, but, to me, this sounds like a very reasonable feature request for Emacs itself. – David J. Jan 1 '15 at 18:46

See also 49.4 The Emacs Initialization File which points to 49.4.4 How Emacs Finds Your Init File. – David J. Jan 1 '15 at 18:49

5 @DavidJames You're right: there is actually a wishlist item about this in Emacs' bug tracker. – Francesco Mar 6 '15 at 23:09

2 Update: it looks like this feature does not interest Emacs developers so much: the request has been tagged `wontfix` and closed in the bug tracker. – Francesco Mar 24 '16 at 20:52

@Francesco: I try a dynamic approach which allows to put the user Emacs directory outside "~". May be you will give a try. – antonio Aug 7 '16 at 18:31

10 Answers

37

The basic approach I use for this is to modify `$HOME`, by running:

```
env HOME=/path/to/dir emacs
```

You then use `/path/to/dir/.emacs.d`

✓

You may wish to also symlink any files or directories of importance in this fake home dir back to the real ones, so that Emacs will see them.

answered Dec 10 '14 at 7:29
phils
29k 2 39 74

2 This works perfect! – Mattias Bengtsson Dec 10 '14 at 8:14

8 According to [Emac Manual - Appendix C Command Line Arguments for Emacs Invocation - Environment Variables - General Variables](), HOME sets "The location of your files in the directory tree; used for expansion of file names starting with a tilde (~)." Changing HOME sounds like a recipe for trouble later when you want to quickly navigate or find files from your *real* home folder. – David J. Jan 1 '15 at 18:43

2 David James: Yes, that's a minor annoyance with this approach. As mentioned, you'll want to copy or symlink things which you *need* Emacs to see under HOME, and if you want to visit your real home directory then you need to use the absolute path (or you could add a symlink to that as well). – phils Jan 1 '15 at 23:28

3 You might also experiment with restoring the original HOME within Emacs during initialisation. I've not tried that, but it seems worth looking into. – phils Jan 1 '15 at 23:32

1 @phils: I [tried]() what you suggest. – antonio Aug 7 '16 at 18:21

38

The way I use to maintain several `.emacs.d` directories in parallel is the following.

1. emacs is started like this:

```
alias emacs='emacs -q --load "/path/to/init.el"'
```

2. Each `init.el` file begins like this, to correctly set up the `user-init-file` and `user-emacs-directory` variables:

```
(setq user-init-file (or load-file-name (buffer-file-name)))
(setq user-emacs-directory (file-name-directory user-init-file))
```

I have found this to work very reliably over the past months. Here are a few remarks:

- it breaks `emacs-init-time`, which only reports the time needed to load the default system configuration, but not your own init file. If you're interested in benchmarking your init time, you'll have to do it another way (see for example [How do I measure performance of elisp code?]()).
- it is not equivalent to a normal startup and you'll need to take care of a few specific points. In particular:
  - `after-init-hook` is run *before* the init file is loaded.
  - The `*scratch*` buffer is created *before* the init file is loaded. You'll have to change its mode explicitly (instead of using `initial-major-mode`).
  - You'll need to explicitly call `package-initialize`; it won't be done automatically
- the path to `init.el` can be arbitrarily chosen; in particular, the directory in which `init.el` resides doesn't have to be named `.emacs.d`. I use this to have for example `.emacs.d.23` alongside `.emacs.d.24` in order to be able to switch between different versions of emacs (the

system I'm using at work is passably outdated, and I can't install emacs 24 on all the machines I use).

- this workflow doesn't require modifying the environment (and especially the `HOME` envvar), which can be desirable if you run programs from within emacs, which could be affected by the modified environment).

edited Apr 13 '17 at 12:52
Community ♦
1

answered Dec 10 '14 at 10:19
Francesco
4,591 18 33

1 This does (in effect) alter the normal execution order, if you are considering the `--load` ed file to be the init file. For starters, it looks to me as if normal (default) package initialisation won't occur, and `after-init-hook` will run *before* the (fake) init file is evaluated. These are things that you can work around, certainly, but be aware that it's not exactly the same thing as Emacs using the specified path as the init file. – phils Dec 10 '14 at 13:31

2 @phils yes, you're right. This does indeed change the normal execution order, and is not equivalent to using a regular init file. I edited my answer to reflect your point on `after-init-hook` . But I have to say that although I use this technique all the time, I never encountered any problem with `after-init-hook` (but I don't use it explicitly, and maybe I'm just lucky that the packages I use don't rely on it). What do you mean by "normal (default) package initialisation won't occur"? – Francesco Dec 10 '14 at 19:57

1 I mean that `command-line` won't call `package-initialize` in that situation. You would need to call it manually in the fake init file. – phils Dec 10 '14 at 22:07

1 @phils thanks. I added this to the answer, along with a mention that the initial major mode must also be taken care of specifically. – Francesco Dec 11 '14 at 7:56

This worked perfectly.. Thank you!!! – Stryker Aug 18 '17 at 5:42

15

You can **symlink** `~/.emacs.d` , this is what I do

1. Try to keep my emacs configuration `~/.emacs.d` oriented i.e. all emacs related config files should live in that folder
2. Then I have an `~/.emacs_configs` folder where all config folders (basically a folder with a `init.el` and rest of the configuration) live, so my personal config folder will be `~/emacs_configs/iqbal` , a prelude distribution will be in `~/emacs_configs/prelude`
3. Very early in my personal emacs config I set the `user-emacs-directory` to the full path to my config using the following

   ```
   (setq user-emacs-directory (file-truename "~/.emacs.d/"))
   ```

4. Then finally I symlink `~/.emacs.d` to the configuration I actually want to use, eg. to use my configuration I will do `ln -s ~/emacs_configs/iqbal .emacs.d` . If you want to tryout some configuration just copy the configuration folder to `~/emacs_configs/whatever_name` and change the symlink

The advantage of the 3rd step is that emacs started with my personal configuration can run unaffected even if I change the `.emacs.d` symlink while emacs it is running.

Another advantage is since the `HOME` is not changed external programs emacs might need to interact with are unaffected

edited Apr 26 '15 at 10:34
AdrieanKhisbe
174 10

answered Dec 17 '14 at 2:43
Iqbal Ansari
6,120 21 30

1 Does this mean we can tweak all the separate emacs configs to `(setq user-emacs-directory (file-truename "~/.emacs.d/"))` so they can all run unaffected simultaneously? – user1011471 Jun 23 '16 at 17:40

1 In theory yes, but in practice there might be some libraries that hard-code the path to `~/.emacs.d` rather than using `user-emacs-directory`. I have come across atleast one such library but unfortunately cannot remember the name. – Iqbal Ansari Jun 24 '16 at 7:19

7

A configuration that doesn't change `HOME` or works with symlinks can be found in my answer https://emacs.stackexchange.com/a/20508/934. With this configuration you can change the `user-emacs-directory` by setting an environment variable:

```
EMACS_USER_DIRECTORY=~/.emacsenv.d/spacemacs emacs
```

and this even works with the daemon.

edited Apr 13 '17 at 12:52
Community ♦
1

answered Feb 24 '16 at 21:47
Uwe Koloska
618 5 11

5

I found this neat solution from EmacsWiki:

```
emacs -q -l ~/my-init-file.el
```

(not exactly using a custom directory, but works nicely because you most likely have a single entry file anyway)

answered Mar 31 '16 at 13:56
phunehehe
159 1 3

FYI, this is redundant with a comment by @Francesco from 2 years ago. – Bryce Sep 14 '18 at 20:42

4

The patch which allows you to specify .emacs.d location via `EMACS_USER_DIRECTORY' environment variable is available in https://debbugs.gnu.org/cgi/bugreport.cgi?bug=15539 but it's not yet merged.

answered Nov 3 '16 at 22:36

god
106 4

3

Set your var **before** loading your init file:

```
emacs -q --eval '(setq alt-conf t)' --load ~/.emacs
```

Then, in your init-file (in this case `~/.emacs`):

```
(defvar alt-conf nil)

(if alt-conf
    (let ((default-directory "~/src/elisp-test/"))
      (normal-top-level-add-subdirs-to-load-path)
      (various-alt-config-stuff)
      (message "Alternate conf"))
  (message "Regular conf"))
```

answered Jul 30 '17 at 19:57
yPhil
465 2 15

Very elegant, my favourite system, since everything remains in Emacs' domain, as it should. Thank you. – gsl Oct 17 '17 at 8:26

1

Expanding on the answer from @phils I made this little shell script (called `testrun.sh`) for testing out my new emacs config. This might make sense to do in other cases as well (for example when testing changes to your init.el that might break emacs).

```
#!/bin/bash

cd $(dirname "${BASH_SOURCE[0]}")
[ -d .testrun ] || mkdir .testrun
cd .testrun
[ -h .emacs.d ] || ln -s .. .emacs.d

env HOME=`pwd` emacs

rm .emacs.d
cd ..
rm -rf .testrun
```

edited Dec 10 '14 at 8:33
phils
29k 2 39 74

answered Dec 10 '14 at 8:17
Mattias Bengtsson
619 1 6 15

Here is a little script based on @Phil's answer and comment about changing the `HOME` environment

1

variable, and then restoring it within Emacs.

```
#!/bin/bash

# Use it like this:
#   /path/to/this/script  EMACS_USER_DIRECTORY  [OTHER EMACS ARGS]

# You can never be too careful
set -e

# First arg = emacs user directory
#   (get a canonical, absolute path)
EMACS_USER_DIRECTORY=$(readlink -f "$1")
shift
if [ ! -d "${EMACS_USER_DIRECTORY}" ]; then
    echo "Non-existent directory: '${EMACS_USER_DIRECTORY}'"
    exit 1
fi

# Bootstrap directory
BOOTSTRAP=$(mktemp --directory --tmpdir .emacs-bootstrap.XXXXXX)
mkdir "${BOOTSTRAP}/.emacs.d"

# Bootstrap init file
cat >"${BOOTSTRAP}/.emacs.d/init.el" <<EOF
  ;; # Correctly set-up emacs-user-directory
  (setq user-emacs-directory "${EMACS_USER_DIRECTORY}/")
  (setq user-init-file (concat user-emacs-directory "init.el"))

  ;; # Reset the HOME environment variable
  (setenv "HOME" "${HOME}")

  ;; # Load the real init file and clean-up afterwards
  (unwind-protect (load user-init-file)
    (delete-directory "${BOOTSTRAP}" :recursive))
EOF

# Forward remaining arguments to emacs
exec env HOME="${BOOTSTRAP}" emacs "$@"
```

answered Mar 24 '16 at 21:21

Francesco
4,591 18 33

1

If the use case is sharing single emacs configuration ".emacs.d" directory across all users of a linux machine then this solution https://emacs.stackexchange.com/a/4258/5488 would work in most cases, but in some cases emacs tries to write temporary files to the user-emacs-directory (such as .ido.last file). In such cases if the shared config directory has write permission to all users then it will work but may not be desired solution as each system user may not want to share the same directory to store temp files. In such case the following solution will be better option.

The common shared config file .emacs.d/init.el should start with

```
;; should come before calling package-initialize as it will populate
;; everything under common config "~/.emacs.d/elpa"
(setq user-init-file (or load-file-name (buffer-file-name)))
(setq package-user-dir (concat (file-name-directory user-init-file) "elpa"))
```

```
(package-initialize)
```

Make the shared config .emacs.d have read permission to all users(need not have write permissions)

```
another_user $ emacs -q --load /path/to/shared/config/.emacs.d/init.el
```

Every user will have his own "~/.emacs.d/" directory but only used to save the temporary files but the packages and other config are loaded from the shared config directory.

answered Jan 2 at 14:43

Talespin_Kit
173 5