

```
(timezone "Europe/Paris")  
(locale "en_US.utf8")
```

```
(bootloader (grub-configuration  
  (device "/dev/sda")))
```

```
(mapped-devices (list (mapped-device  
  (source "/dev/sda3")  
  (target "home")  
  (type luks-device-mapping))))
```

Composing system abstractions with GNU Guix

```
(file-systems (cons* (file-system  
  (device "root")  
  (title "label")  
  (mount-point "/")  
  (type "ext3"))  
  (file-system  
    (device "/dev/mapper/home")  
    (mount-point "home"))
```

Ludovic Courtès

FOSDEM, 7 February 2021

Desired state and 'idempotency'

Most Ansible modules **check whether the desired final state has already been achieved**, and exit without performing any actions if that state has been achieved, so that repeating the task does not change the final state. Modules that behave this way are often called 'idempotent.' Whether you run a playbook once, or multiple times, the outcome should be the same. However, **not all playbooks and not all modules behave this way**. If you are unsure, test your playbooks in a sandbox environment before running them multiple times in production.

```
{ fetchurl, stdenv } : ← function definition
```

```
stdenv . mkDerivation {  
  name = "hello-2.3";  
  src = fetchurl {  
    url = mirror://gnu/hello/hello-2.3.tar.bz2;  
    sha256 = "0c7vijq8y68...";  
  };  
};
```

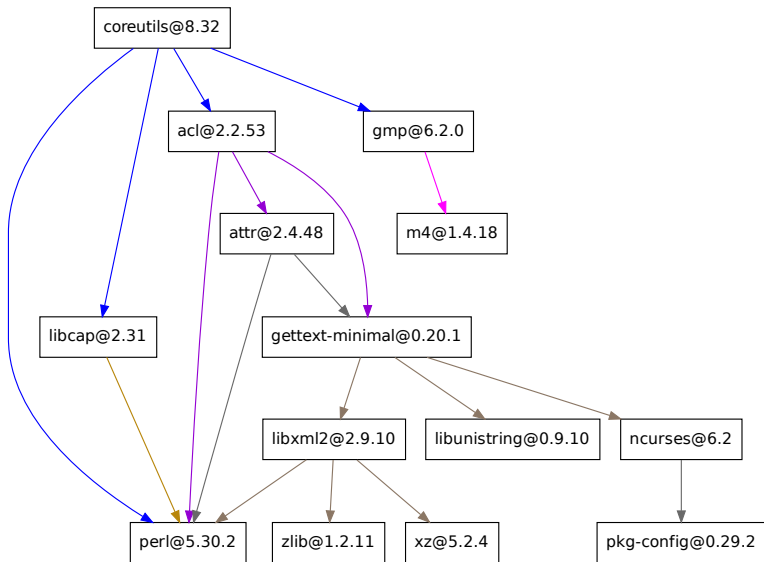
function call

```
meta = {  
  description = "Produce a friendly greeting";  
  homepage = http://www.gnu.org/software/hello/  
  license = "GPLv3+";  
};  
}
```

Bill of rights:

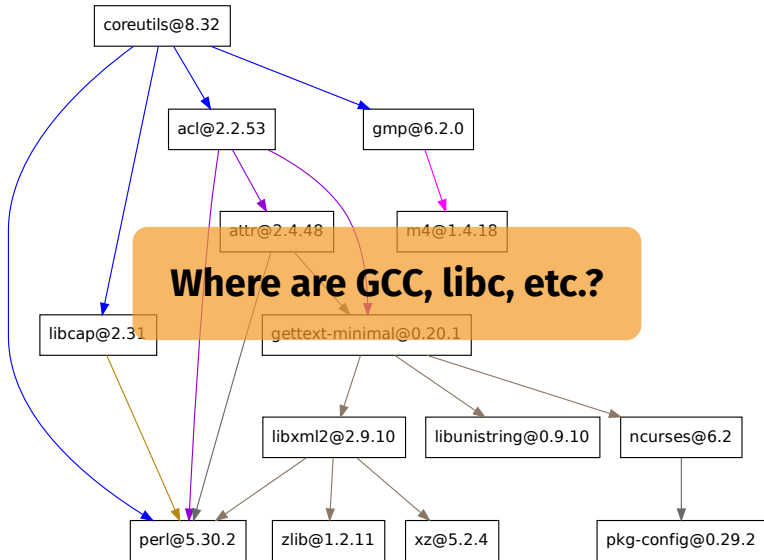
1. What You Declare is What You Get
2. Declarations Bring Structure

The package structure.



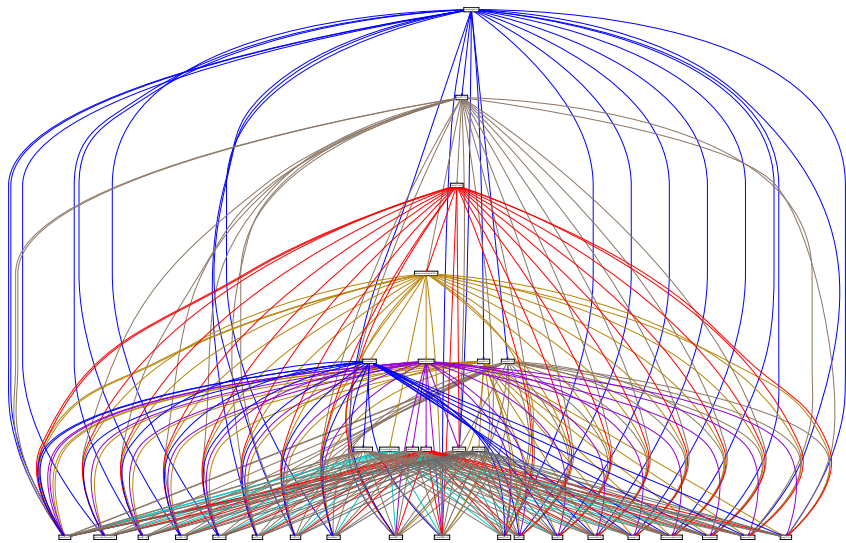
14 nodes

guix graph --type=package coreutils



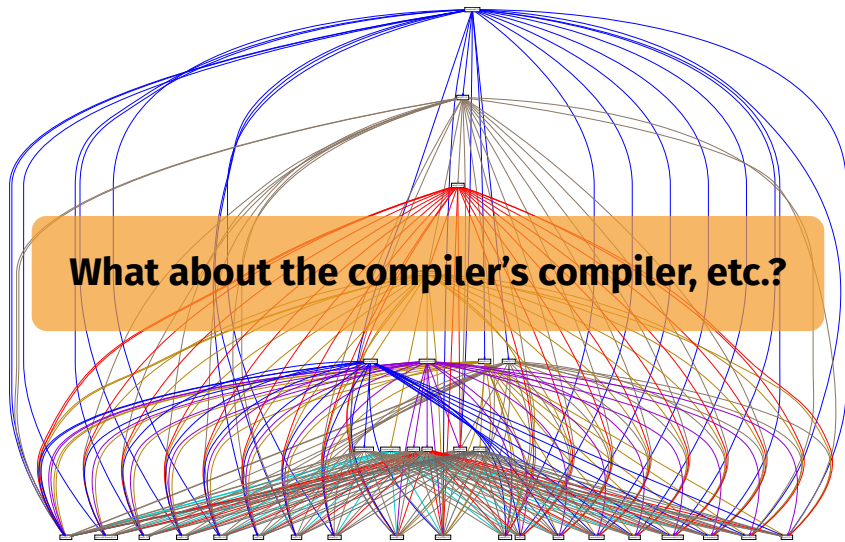
14 nodes

guix graph --type=package coreutils



33 nodes

guix graph --type=bag-emerged coreutils



33 nodes

`guix graph --type=bag-emerged coreutils`

(too big)

120 nodes

```
guix graph --type=bag coreutils
```

(too big)

455 nodes

```
guix graph --type=derivation coreutils
```

```
(define audacity
  (package
    (name "audacity")
    (home-page "https://github.com/audacity/audacity")
    (source (origin
              (method git-fetch)
              (uri (git-reference
                    (url home-page)
                    (commit "2f30ff07a")
                    (recursive? #t)))
              (sha256
               (base32
                "106rf402cvfdhc2yf...")))))
  ...))
```

```
(define audacity
  (package
    (name "audacity")
    (home-page "https://github.com/audacity/audacity")
    (source (origin
      (method git-fetch)
      (uri (git-reference
        (url home-page)
        (commit "2f30ff07a")
        (recursive? #t)))
      (sha256
        (base32
          "106rf402cvfdhc2yf...")))))
  ...))
```



Software Heritage



```
guix lint -c archival audacity
```

```
$ guix install gcc-toolchain openmpi hwloc
```

```
...
```

hint: Consider setting the necessary environment variables by running:

```
GUIX_PROFILE="$HOME/.guix-profile"  
. "$GUIX_PROFILE/etc/profile"
```

Alternatively, see ‘guix package --search-paths’.

```
(define gcc
  (package
    ;; ...
    (native-search-paths
      (list (search-path-specification
              (variable "C_INCLUDE_PATH")
              (files '("include"))))
            (search-path-specification
              (variable "CPLUS_INCLUDE_PATH")
              (files '("include/c++" "include")))
            (search-path-specification
              (variable "LIBRARY_PATH")
              (files '("lib" "lib64"))))))))
```

```
guix pack hwloc \  
  --with-source=./hwloc-2.1rc1.tar.gz
```

```
guix install emacs-next \  
  --with-branch=emacs-next=master
```

```
guix build guix \  
  --with-latest=guile-json
```


- ▶ `--with-patch`
- ▶ `--with-input`
- ▶ `--with-c-toolchain`
- ▶ `--with-debug-info`
- ▶ `--with-graft`
- ▶ ...

```
(define (make-glibc-utf8-locales locales)
  (package
    (name "glibc-utf8-locales")
    ...))
```

```
(define glibc-utf8-locales
  (make-glibc-utf8-locales (list "ca_ES.utf8" ...)))
```

```
(define emacs  
  (package  
    (name "emacs")  
    ...))
```

```
(define emacs-no-x  
  (package/inherit emacs  
    (name "emacs-no-x")  
    ...))
```

```
(define emacs-xwidgets  
  (package/inherit emacs  
    (name "emacs-xwidgets")  
    ...))
```

```
(define emacs  
  (package  
    (name "emacs")  
    ...))
```

```
(define emacs-no-x  
  (package/inherit emacs  
    (name "emacs-no-x")  
    ...))
```

```
(define emacs-xwidgets  
  (package/inherit emacs  
    (name "emacs-xwidgets")  
    ...))
```

What if we could declare **package parameters**?

```
(define emacs  
  (package  
    (name "emacs")  
    ...))
```

```
(define emacs-no-x  
  (package/inherit emacs  
    (name "emacs-no-x")  
    ...))
```

What if we could declare **package parameters**

```
(define emacs-xwidgets  
  (package/inherit emacs  
    (name "emacs-xwidgets")  
    ...))
```



WIP!

Package collections.

```
guix package --manifest=my-packages.scm
```

```
(specifications->manifest  
  '("emacs" "emacs-geiser"  
    "guile" "guile-picture-language"))
```

```
guix install emacs
```

```
guix install guile guile-picture-language
```

```
guix install emacs-geiser
```


guix **install** emacs

guix **install** guile guile-picture-language
guix package **--export-manifest**

guix **install** emacs-geiser



New!

Systems.

```
(operating-system
  (host-name "guixbox")
  (timezone "Europe/Brussels")
  (locale "fr_BE.utf8")
  (bootloader (bootloader-configuration
    (bootloader grub-efi-bootloader)
    (target "/boot/efi")))
  (file-systems (append (list (file-system
    (device (file-system-label "my-root"))
    (mount-point "/")
    (type "ext4")))
    %base-file-systems))
  (users (append (list (user-account
    (name "charlie")
    (group "users")
    (home-directory "/home/charlie")))
    %base-user-accounts))
  (services (append (list (service dhcp-client-service-type)
    (service openssh-service-type))
    %base-services)))
```

```

(operating-system
  (host-name "guixbox")
  (timezone "Europe/Brussels")
  (locale "fr_BE.utf8")
  (bootloader (bootloader-configuration
    (bootloader grub-efi-bootloader)
    (target "/boot/efi")))
  (file-systems (append (list (file-system
    (device (file-system-label "my-root"))
    (mount-point "/" )
    (type "ext4"))))
    %base-file-systems))
  (users (append (list (user-account
    (name "charlie")
    (group "users")
    (home-directory "/home/charlie"))))
    %base-user-accounts))
  (services (append (list (service dhcp-client-service-type)
    (service openssh-service-type))
    %base-services)))

```

guix system vm config.scm

```
(operating-system
  (host-name "guixbox")
  (timezone "Europe/Brussels")
  (locale "fr_BE.utf8")
  (bootloader (bootloader-configuration
    (bootloader grub-efi-bootloader)
    (target "/boot/efi")))
  (file-systems (append (list (file-system
    (device (file-system-label "my-root"))
    (mount-point "/")
    (type "ext4"))))
    %base-file-systems))
  (users (append (list (user-account
    (name "charlie")
    (group "users")
    (home-directory "/home/charlie"))))
    %base-user-accounts))
  (services (append (list (service dhcp-client-service-type)
    (service openssh-service-type))
    %base-services)))
```

guix system **docker-image** config.scm

```

(operating-system
  (host-name "guixbox")
  (timezone "Europe/Brussels")
  (locale "fr_BE.utf8")
  (bootloader (bootloader-configuration
    (bootloader grub-efi-bootloader)
    (target "/boot/efi")))
  (file-systems (append (list (file-system
    (device (file-system-label "my-root"))
    (mount-point "/" )
    (type "ext4"))
    %base-file-systems))
  (users (append (list (user-account
    (name "charlie")
    (group "users")
    (home-directory "/home/charlie"))
    %base-user-accounts))
  (services (append (list (service dhcp-client-service-type)
    (service openssh-service-type))
    %base-services)))

```

guix system **container** config.scm

```
(operating-system
  (host-name "guixbox")
  (timezone "Europe/Brussels")
  (locale "fr_BE.utf8")
  (bootloader (bootloader-configuration
    (bootloader grub-efi-bootloader)
    (target "/boot/efi")))
  (file-systems (append (list (file-system
    (device (file-system-label "my-root"))
    (mount-point "/" )
    (type "ext4"))
    %base-file-systems))
  (users (append (list (user-account
    (name "charlie")
    (group "users")
    (home-directory "/home/charlie"))
    %base-user-accounts))
  (services (append (list (service dhcp-client-service-type)
    (service openssh-service-type))
    %base-services)))
```

guix system reconfigure config.scm

(operating-system

```
(host-name "guibox")
```

```
(timezone "Europe/Brussels")
```

```
(locale "fr_BE.utf8")
```

```
(bootloader (bootloader-configuration
```

```
  (bootloader grub-loader)
```

```
  (target "/boot"))
```

```
(file-systems (append (list (file-system
```

```
  (device (file-system-label "my-root"))
```

```
  (mount-point "/")
```

```
  (type "ext4"))
```

```
  %base-file-systems))
```

```
(users (append (list (user-account
```

```
  (name "charlie")
```

```
  (group "users")
```

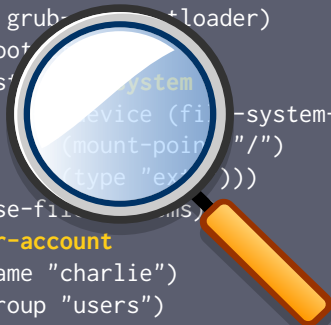
```
  (home-directory "/home/charlie"))
```

```
  %base-user-accounts))
```

```
(services (append (list (service dhcp-client-service-type)
```

```
  (service openssh-service-type))
```

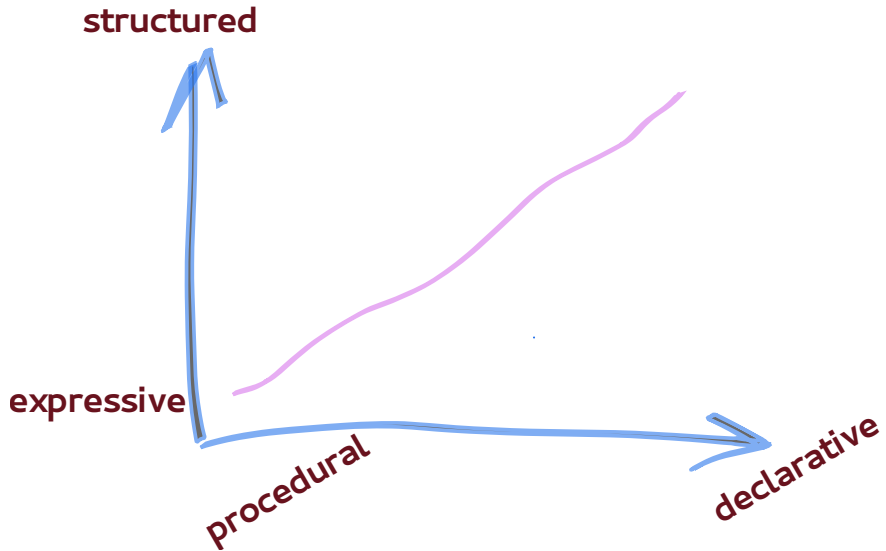
```
  %base-services)))
```





<https://notabug.org/civodul/guix-explorer>

Wrap-up.





Guix

<https://guix.gnu.org/>

ludo@gnu.org

Copyright © 2010, 2012–2021 Ludovic Courtès ludo@gnu.org.

GNU Guix logo, CC-BY-SA 4.0, <https://gnu.org/s/guix/graphics>.

FIXME Magnifying glass picture © FIXME https://commons.wikimedia.org/wiki/File:Magnifying_glass_CC0.svg

Copyright of other images included in this document is held by their respective owners.

This work is licensed under the **Creative Commons Attribution-Share Alike 3.0** License. To view a copy of this license, visit <https://creativecommons.org/licenses/by-sa/3.0/> or send a letter to Creative Commons, 171 Second Street, Suite 300, San Francisco, California, 94105, USA.

At your option, you may instead copy, distribute and/or modify this document under the terms of the **GNU Free Documentation License, Version 1.3 or any later version** published by the Free Software Foundation; with no Invariant Sections, no Front-Cover Texts, and no Back-Cover Texts. A copy of the license is available at <https://www.gnu.org/licenses/gfdl.html>.

The source of this document is available from <https://git.sv.gnu.org/cgit/guix/maintenance.git>.