Installer GuixSD 0.16 à côté de Debian Testing/Sid

- Objet : Installation en Dual Boot
- Niveau requis :
  avisé

  L'installation de **Guix System 1.0.1** ne devrait pas être très différente à effectuer (non testé).

- Commentaires :
  https://www.gnu.org/software/guix/manual/html_node/GNU-Free-Documentation-License.html
- Débutant, à savoir : https://www.gnu.org/software/guix/manual/fr/guix.fr.html

Avant l'installation

(Chez moi, l'ordinateur utilisé est un portable “**Packard Bell EasyNote TE** “Intel Celeron B830. Intel Graphics Media Accelerator (GMA) HD Graphics)

Téléchargement de GNU Guix Software Distribution : https://www.gnu.org/software/guix/download/

Copie de l'image sur une clé USB

Pour copier l'image sur une clé USB, j'ai suivi ces étapes:

1. Décompresser l'image en utilisant la commande xz:

```
$ xz -d guixsd-install-0.16.0.x86_64-linux.iso.xz
```

2. Insérer une clé USB d'au moins 1,2 GiB dans votre machine, et déterminer son nom de périphérique. Si la clé USB est connue comme /dev/sdb, copier l'image avec:

```
# dd if=guixsd-install-0.16.0.x86_64-linux.iso of=/dev/sdb
```

```
# sync
```

L'accès à /dev/sdX requiert habituellement les privilèges de root.

Graver l'image sur un DVD

Pour graver l'image sur un DVD, suivez ces étapes:

1. Décompresser l'image en utilisant la commande xz:

```
$ xz -d guixsd-install-0.16.0.x86_64-linux.iso.xz
```

2. Insérer un DVD vierge et déterminer son nom de périphérique. Si le nom du périphérique DVD est connu comme /dev/srX, copier l'image avec:

```
# growisofs -dvd-compat -Z /dev/srX=guixsd-install-0.16.0.x86_64-linux.iso
```

Sources :

https://www.gnu.org/software/guix/manual/html_node/Preparing-for-Installation.html#Preparing-for-Installation

Il est **hautement recommandé** de télécharger et graver **Supergrub2disk** (ou bien de le copier sur une clé USB), spécialement dans ces circonstances d'une installation en dual-boot avec Debian Testing/Sid.

Cela peut réellement vous aider... Moi, les 2 fois où j'en ai eu besoin, il m'a servi avec succès.
https://www.supergrubdisk.org/category/download/supergrub2diskdownload/

Installation

L'image d'installation utilise la disposition du clavier qwerty.

Si je veux le changer, je peux utiliser la commande loadkeys. Par exemple, la commande suivante sélectionne la disposition du clavier fr, donc allons-y :

```
# loadkeys fr
```

Je ne vais pas aborder la question du **Wi-Fi** ici, car ma connexion est très lente (Zone non dégroupée).

Cependant, le matériel ne m'impose aucune restriction (adaptateur réseau sans fil **Qualcomm Atheros AR9485 (rév. 01)**.

Pour plus d'informations sur le Wi-Fi, voir ci-dessous :

https://www.gnu.org/software/guix/manual/html_node/Preparing-for-Installation.html#Networking

Certaines cartes wifi nécessitent l'utilisation d'un firmware non libre : pour un **autre ordinateur portable**, j'avais dû commander un adaptateur USB sans fil **Technoethical N150 Mini** pour GNU/Linux-libre à l'adresse :

https://tehnoetic.com/adapters/tehnoetic-wireless-adapter-gnu-linux-libre-tet-n150

Pour cette installation, j'ai donc utilisé la connexion d'un joli câble RJ45.

```
# ifconfig -a
```

Les interfaces filaires ont un nom commençant par ‘e’; Par exemple, l’interface correspondant au premier contrôleur Ethernet intégré est appelée «eno1».

```
# ifconfig enp2s0f0 up
```

À ce stade, je dois faire l'acquisition d'une adresse IP. Les adresses IP étant automatiquement attribuées via DHCP, je peux exécuter :

```
# dhclient -v enp2s0f0
```

Essayons d’envoyer une requête ping à un serveur pour voir si la mise en réseau est opérationnelle : (sachant que j'habite dans une zone non dégroupée, vitesse 2.0 Mbs)

```
# ping -c 3 gnu.org
PING gnu.org (208.118.235.148): 56 data bytes
64 bytes from 208.118.235.148: icmp_seq=0 ttl=51 time=138.917 ms
64 bytes from 208.118.235.148: icmp_seq=1 ttl=51 time=139.388 ms
64 bytes from 208.118.235.148: icmp_seq=2 ttl=51 time=138.998 ms
--- gnu.org ping statistics ---
3 packets transmitted, 3 packets received, 0% packet loss
round-trip min/avg/max/stddev = 138.917/139.101/139.388/0.206 ms
```

Partitions

Partitionnement **GPT ou EFI** : Les tests que j'ai effectués dans les machines virtuelles n'ayant pas été très concluants, je n'en sais pas trop sur le sujet. Si vous souhaitez utiliser GPT ou EFI, veuillez vous reporter à la documentation :

https://www.gnu.org/software/guix/manual/html_node/Preparing-for-Installation.html#Disk-Partitioning

Dans mon cas, sachant que je n'aurai pas à m'inquiéter de cette question pour cause d'ordinateur datant de 2012, c'est parti :

```
# cfdisk
```

```
Device     Boot     Start     End         Sectors    Size   Id Type
/dev/sda1            2048    532542415   532540368    254G  83 Linux
/dev/sda2       532543486    976771071   444227586  211.8G   5 Extended
  /dev/sda5     968720384    976771071   8050688      3.9G  82 Linux swap /
Solaris
```

```
  /dev/sda6 *   532543488    968720383   436176896    208G  83 Linux
Free space      976771072    976773167        2096      1M
```

GuixSD sera installé sur /dev/sda6.

Concernant votre propre installation, il est préférable d'activer l'option de démarrage * sur votre /dev/sdX maintenant…

Attribuez une étiquette aux systèmes de fichiers afin de pouvoir les consulter facilement et de manière fiable dans les déclarations de système de fichiers.

Cela se fait généralement à l'aide de l'option **-L** de **mkfs.ext4** et des commandes associées. En supposant que la partition racine cible se trouve dans /dev/sda6, vous pouvez créer un système de fichiers portant le libellé **my-root** avec la commande suivante. Personnellement, j'ai fait :

```
# mkfs.ext4 -L my-root /dev/sda6
```

Monter le système de fichiers cible sous /mnt avec cette commande (encore une fois, en supposant que **my-root** est l’étiquette du système de fichiers racine) :

```
# mount LABEL=my-root /mnt
```

J'ai une partition de swap à disposition sur /dev/sda5 pour Debian, pourquoi ne pas l'utiliser ? J'ai lancé :

```
# mkswap /dev/sda5
```

```
# swapon /dev/sda5
```

C'est parti !

Avec les partitions cibles prêtes et la racine cible montée sur /mnt, on est bon. Premièrement :

```
# herd start cow-store /mnt
```

Cela crée une copie en écriture de **/gnu/store**, de telle sorte que les packages qui lui sont ajoutés au cours de la phase d’installation soient écrits sur le disque cible sur **/mnt** plutôt que conservés en mémoire.

Cela est nécessaire car la première phase de la commande **guix system init** implique le téléchargement ou la compilation dans /gnu/store, qui est initialement un système de fichiers en mémoire.

Création du répertoire cible de configuration du système :

```
# mkdir /mnt/etc
```

Création du prochain fichier de configuration système (vide pour l'instant) :

```
# touch /etc/configuration/perso.scm
```

Configuration en cours d'installation

Vous avez également la possibilité d'éditer un fichier existant déjà dans ***/etc/configuration*** (par exemple, "**desktop.scm**" ou "**bare-bones.scm**" et le modifier à votre guise).

```
# vi /etc/configuration/desktop.scm
```

```
# vi /etc/configuration/bare-bones.scm
```

(**Nano** et **Zile** sont toutefois disponibles).

```
# vi /etc/configuration/perso.scm
```

/etc/configuration/perso.scm

```
;; Operating system configuration for a full
;; "desktop" environment with gnome.

(use-modules (gnu) (gnu system nss))
;; Le fichier est en langage guile scheme. Après
(use-service-modules desktop networking ssh)
;; un peu d'utilisation, sa pratique est plus aisée.
(use-package-modules bootloaders certs gnome screen ssh)
;; Module 'bootloaders' indispensable pour le Dual-boot.

(operating-system
  (host-name "gnu")
  (timezone "Europe/Paris")
  (locale "fr_FR.utf8")

;; En admettant que /dev/sda est la cible du disque dur, et "my-root"
est le nom de la cible root du système de fichiers,
;; et /dev/sda6 la partition cible pour l'installation de GuixSD en
Dual-Boot.
;; La partition /dev/sda6 sera détectée par le système, pour cela, elle
ne doit pas figurer dans le fichier.
;; Si vous voulez juste installer GuixSD, (bootloader (grub-
configuration (target "/dev/sda") sera suffisant.
(bootloader (grub-configuration (target "/dev/sda") (menu-entries
;; Ajout de menu-entries
                  (list                                          ;;
Liste d'entrées
            (menu-entry                                  ;; Entrons
```

```
                (label "Debian GNU/Linux")                          ;;
Titre de la distribution
                (linux "/boot/vmlinuz-4.19.0-1-amd64")          ;;
Noyau à démarrer
                (linux-arguments '("root=/dev/sda1"))           ;;
Debian se trouve sur /dev/sda1
                (initrd "/boot/initrd.img-4.19.0-1-amd64"))))))  ;;
C'est parti pour le démarrage !

(file-systems (cons (file-system
                      (device (file-system-label "my-root"))
                      (mount-point "/")
                      (type "ext4"))
                    %base-file-systems))

(swap-devices '("/dev/sda5"))

(users (cons (user-account
              (name "hubert")
              (comment "GuixSD user")
              (group "users")
              (supplementary-groups '("wheel" "netdev"
                                      "audio" "video"))
              (home-directory "/home/hubert"))
             %base-user-accounts))

;; This is where we specify
;; system-wide packages.
(packages (cons* gvfs                   ;; for users mounts
                 nss-certs              ;; for HTTPS access
                 screen
                 openssh
                %base-packages))

;; **Edit** 19-04-2019 : 'gnome-desktop-service est déprécié depuis
quelques jours.
;; Cela devrait marcher si vous remplacez (gnome-desktop-service) par
(service gnome-desktop-service-type)
(services (cons* (service gnome-desktop-service-type)  ;; Testé, c'est
bon.
                 %desktop-services))

;; Allow resolution of '.local' host names with mDNS.
(name-service-switch %mdns-host-lookup-nss))
```

Faisons maintenant une copie de ce fichier renommé '**config.scm**' sur '**/mnt/etc**'

```
# cp /etc/configuration/perso.scm /mnt/etc/config.scm
```

Une fois la préparation du fichier de configuration terminée, le nouveau système doit être initialisé (rappelez-vous que le système de fichiers racine cible est monté sur **/mnt**) :

```
# guix system init /mnt/etc/config.scm /mnt
```

Ceci copie tous les fichiers nécessaires et installe GRUB sur /dev/sda (à moins que vous ne passiez l'option *–no-bootloader* .

Cette commande peut déclencher des téléchargements ou des versions des packages manquants, ce qui peut prendre un certain temps.

La copie sur /mnt va se terminer…

[##########################]

L'installation est terminée, on peut redémarrer :

```
# reboot
```

Et voilà

Utilisation

Ctrl-Alt-F2 pour basculer en console.

root n'a pas encore de mot de passe mais il est déjà actif.

Au prompt (à l'invite de commande), taper **root** et appuyer sur Entrée.

Un mot de passe pour l'utilisatrice ou l'utilisateur :

```
# passwd hubert
New password:
Retype new password:
passwd: password updated successfully
```

Le mot de passe root peut être créé plus tard…

```
# passwd root
```

Consoles TTY

Lors de la première session d'accès utilisateur, j'ai constaté que je ne pouvais pas aller en pleine console. C'était à cause du bios, il a été nécessaire d'activer les consoles TTY dans l'onglet “Principal” 'Comportement des touches de fonction' [Touches de fonction] Maintenant, Ctrl-Alt-F1 me connecte en pleine console, Ctrl-Alt-F2 aussi, et Ctrl-Alt-F3, Ctrl-Alt-F4 … etc. **Alt-F7** fonctionne également.

Clavier AZERTY

En console, le compte root dispose du clavier azerty. Quant-à ma session GNOME, elle était encore en QWERTY, j'ai dû configurer (graphiquement) “Tous les paramètres” / “Région et langue”, où j'ai ajouté Langue “Français” et dans “Sources d'entrée”: “Français (alt.)”.

Il ne me restait plus qu'à choisir “fr” dans la barre Gnome-Shell du bureau et le clavier est devenu AZERTY.

Mises-à-jour

A faire à peu près une fois par semaine

```
$ guix pull
```

```
$ guix package -u
```

```
$ sudo guix system reconfigure /etc/config.scm
```

```
# reboot
```

Remarque : Lors d'une mise-à-jour, la commande **guix package -u** effectue l'opération intégralement. Par exemple, chez moi le paquetage *webkitgtk* met très longtemps à se construire sous forme de dérivation. Idem pour *ungoogled-chromium*. Pour gagner du temps, j'ai employé la commande suivante :

```
$ guix package --upgrade . --do-not-upgrade webkitgtk ungoogled-chromium
```

La mise-à-jour s'est effectuée beaucoup plus rapidement et, un redémarrage plus tard, Chromium se lance et fonctionne parfaitement bien.

Bien entendu, il arrivera un moment où la mise-à-jour intégrale redeviendra nécessaire. Il faudra à nouveau faire preuve de patience...

À moins que... l'écriture d'un fichier 'manifest' me permette de confiner mon profil logiciels de manière satisfaisante. En effet les mises-à-jour depuis un fichier manifest sont beaucoup plus rapides. Reste à mettre en place cette pratique... **Infos :** https://www.gnu.org/software/guix/manual/fr/html_node/Invoquer-guix-package.html#profile_002dmanifest

Installation de logiciels

Avec GuixSD, les installations de logiciels peuvent se faire depuis le compte user.

```
$ guix package -i markdown
```

markdown convertit un ficher texte en fichier html en une seule ligne de commande.

```
$ guix package -i ungoogled-chromium
```

ungoogled-chromium me semble plus rapide et plus abouti que Icecat.

```
$ guix package -i mpv
```

mpv est un fork de mplayer. Légereté et fluidité à l'utilisation.

Pour ce qui est de ***vim***, j'ai préféré l'installer depuis le compte root.

```
# guix package -i vim
```

Suite à ces installations, le système conseille de lancer :

```
$ export PATH="$HOME/.config/guix/current/bin:$PATH"
```

```
$ export INFOPATH="$HOME/.config/guix/current/share/info:$INFOPATH"
```

Source :

https://www.gnu.org/software/guix/manual/en/html_node/Invoking-guix-pull.html#Invoking-guix-pull

Désinstallation de logiciels

Exemples :

```
# guix package -r vim
```

```
$ guix package -r vlc
```

Utilisation de manifest

“Pour gérer le profil de l'utilisat(ric(e)ur, de manière **déclarative**, comme des packages” (**iyzsong**) 😃

Il s'agit de confiner l'environnement logiciel dans le profil personnel. Les mises-à-jour se feront beaucoup plus rapidement, car elles seront relayées depuis un fichier **manifest.scm** qui sera créé dans mon ordinateur, plus exactement dans **~/.config/guix/**

Sources : https://ambrevar.xyz/guix-advance/

En voici les différentes étapes :

1 - Récupérer le script Guile suivant et l'enregistrer à la racine du dossier personnel sous le nom de **manifest-to-manifest.scm**.

manifest-to-manifest.scm

```
;; Run with:
;;     guile -s FILE ~/.guix-profile

(use-modules (guix profiles)
         (ice-9 match)
         (ice-9 pretty-print))

(define (guix-manifest where)
  (sort (map (lambda (entry)
               (let ((out (manifest-entry-output entry)))
                 (if (string= out "out")
                 (manifest-entry-name entry)
                 (format #f "~a:~a"
                     (manifest-entry-name entry)
                     (manifest-entry-output entry)))))
             (manifest-entries (profile-manifest where)))
    string<?))

;; Thanks to Ivan Vilata-i-Balaguer for this:
(define (guix-commit)
  (let ((guix-manifest (profile-manifest (string-append (getenv "HOME")
"/.config/guix/current"))))
    (match (assq 'source (manifest-entry-properties (car (manifest-
entries guix-manifest))))
      (('source ('repository ('version 0) _ _
                 ('commit commit) _ ...))
       commit)
      (_ #f))))

(match (command-line)
  ((_ where)
   (format #t ";; commit: ~a\n" (guix-commit))
   (pretty-print
    `(specifications->manifest
      ',(guix-manifest where))))
  (_ (error "Please provide the path to a Guix profile.")))
```

2 - Appeler le script :

```
$ guile -s manifest-to-manifest.scm ~/.guix-profile
```

Sortie de la commande

```
;;; note: auto-compilation is enabled, set GUILE_AUTO_COMPILE=0
;;;       or pass the --no-auto-compile argument to disable.
;;; compiling /home/hubert/manifest-to-manifest.scm
;;; compiled /home/hubert/.cache/guile/ccache/2.2-
LE-8-3.A/home/hubert/manifest-to-manifest.scm.go
;; commit: 6c.....................................a
(specifications->manifest
  '("brasero"
    "ffmpeg"
    "gimp"
    "gnumeric"
    "inkscape"
    "lynx"
    "markdown"
    "mpv"
    "obs"
    "qemu"
    "quassel"
    "racket"
    "sbcl-next"
    "ungoogled-chromium"
    "wget"
    "youtube-dl"
    "youtube-viewer"))
```

3 - Création de ce nouveau fichier scheme (personnellement avec gedit) :

```
$ gedit
```

~/.config/guix/manifest.scm

```
(specifications->manifest
  '("brasero"
    "ffmpeg"
    "gimp"
    "gnumeric"
    "inkscape"
    "lynx"
    "markdown"
    "mpv"
    "obs"
    "qemu"
    "quassel"
    "racket"
    "sbcl-next"
    "ungoogled-chromium"
    "wget"
    "youtube-dl"
```

```
    "youtube-viewer"))
```

Je le nomme **manifest.scm** et l'enregistre dans **~/.config/guix/**

4 - Arrive l'étape d'installation, de mise-à-jour, de compilation et de construction :

```
$ guix package -m ~/.config/guix/manifest.scm
```

Et voilà un joli fichier manifest tout neuf.

Edit 29 juillet 2019 : Sur la liste de diffusion

Help-Guix j'ai reçu une réponse qui m'a permis d'adapter la syntaxe de la mise-à-jour.

“L'option -u prend un argument optionnel, un regexp correspondant aux packages sur lesquels on agit. Il est préférable de passer explicitement en `-u` dans le cas où cette commande est lancée depuis un “shell history”. On peut alors juste ajouter n'importe quelle option voulue, sans qu'elles soit interprêtée comme argument vers -u.”

Ce qui donne :

```
$ guix pull && guix package -m ~/.config/guix/manifest.scm -u
```

(Merci à **Josh Holland** \o/)

Viennent ensuite les habituelles commandes :

```
$ sudo guix system reconfigure /etc/config.scm
```

```
# reboot
```

Mise-à-jour effectuée bien plus rapidement, ce qui ne va pas être négligeable, habitant en zone non dégroupée.

manifest, c'est bon... **/Edit**

SSH

J'ai mis mon accès ssh pour ovh dans un signet **Fichiers** (Nautilus) et ça fonctionne.

Multimédia

Téléchargement d'une vidéo :

```
$ youtube-dl https://www.youtube.com/watch?v=j7sTHoeH0eA
```

Lecture d'une vidéo en streaming :

```
$ youtube-viewer https://www.youtube.com/watch?v=OSWKw15rCoI
```

```
$ mpv https://www.youtube.com/watch?v=-mfladpK0AA
```

Redéfinir le timing de début et de fin d'une vidéo :

```
$ ffmpeg -i test.mp4 -c copy -ss 00:01:23 -to 00:04:56 cut.mp4
```

Redéfinir le timing de début et de fin d'un morceau audio :

```
$ ffmpeg -i test.mp3 -c copy -ss 00:01:23 -to 00:04:56 cut.mp3
```

Conversion d'un fichier .mkv en **.mp4** :

```
$ ffmpeg -i film-a-convertir.mkv -ab 128k -ac 2 -vcodec libx264 notre-produit-final.mp4
```

Lecture avec :

```
$ mpv notre-produit-final.mp4
```

Convertir mp4 en webm :

```
$ ffmpeg -i input-file.mp4 -c:v libvpx -crf 10 -b:v 1M -c:a libvorbis output-file.webm
```

Ecouter France Musique en direct :

```
$ mpv https://icecast.radiofrance.fr/francemusique-midfi.mp3?id=radiofrance
```

Convertir un mp4 en mp3 :

```
$ ffmpeg -i video.mp4 -vn -ab 192k -acodec libmp3lame -ac 2 audio.mp3
```

MACHINE ViRTUELLE

L'exemple ci-dessous crée une machine virtuelle dans laquelle le répertoire de base de l'utilisateur est accessible en lecture seule et où le répertoire /exchange est un mappage en lecture écriture de $HOME/tmp sur l'hôte :

```
guix system vm my-config.scm \
   --expose=$HOME --share=$HOME/tmp=/exchange
```

Self-reproducing live USB

(**Image disque**)

Extraits traduits de https://ambrevar.xyz/guix-advance/#orgea240b0 :

“Lors de l'utilisation d'une image disque, une image disque brute est produite.

Elle permet par exemple de régénérer le système actuel sur une clé USB.

Cela permet assez facilement la création d'un clone à emporter de son système actuel, qui peut être branché n'importe où et reproduire son environnement informatique exact (sans le matériel).

On peut inclure des données personnalisées telles que ses clés PGP et tout, y compris les courriels, facilement disponibles dès le démarrage.

En outre, il est évidemment possible d’installer le système par clonage sur la machine sur laquelle est branchée la clé USB: au lieu d’être une installation simple de Guix, elle déploie tout votre système d’exploitation personnalisé”.

En supposant que /dev/sdc soit le périphérique correspondant a une clé USB, vous pouvez y copier l image à l aide de la commande suivante :

```
# dd if=$(guix system disk-image my-os.scm) of=/dev/sdc
```

Maintenance

Les 2 commandes suivantes ont déjà été évoquées plus haut, mais je pense qu'elles ont également leur place ici. Elles sont à lancer peu après l'installation du système, celui-ci nous en informant en temps voulu :

```
$ export PATH="$HOME/.config/guix/current/bin:$PATH"
```

```
$ export INFOPATH="$HOME/.config/guix/current/share/info:$INFOPATH"
```

Avoir la garantie que 5 Go vont rester disponibles. Cette commande peut être lancée régulièrement :

```
guix gc -F 5G
```

Suppression des générations vieilles de plus d'un mois :

```
$ guix package --delete-generations=1m
```

Une fois que ceci a été fait, une manière radicale de gagner davantage de place consiste à supprimer les fichiers inutiles en lançant le garbage collector (ramasse-miettes) sans arguments.

```
$ guix gc --collect-garbage
.....................................................................
deleting `/gnu/store/trash'
deleting unused links...
note: currently hard linking saves 8874.56 MiB
guix gc: 5043,55469 Mo libérés*
```

Certaines reconstructions et téléchargements ultérieurs pourront alors être plus longs que prévus.

Il existe d'autres manières de faire plus fines, mais je ne les maîtrise pas pour l'instant...

```
$ man guix gc
```

Aller un peu plus loin

Il peut être intéressant d'automatiser la mise-à-jour de la base de données de l'index du compte user ainsi que les procédures de “garbage collector” énumérées plus haut. Pour celà, il faut ajouter quelques modules au début du fichier de configuration de la machine. Puis insérer 3 définitions :

/etc/config.scm

```
;; Operating system configuration for a full
;; "desktop" environment with gnome

(use-modules (gnu) (guix) (gnu services mcron) (gnu system nss))
;; Ajout de (guix) et de (gnu services mcron)
(use-service-modules desktop networking ssh)
(use-package-modules bootloaders certs gnome screen ssh base idutils)
;; Ajout de base et de idutils

(define updatedb-job
;;  ;; Run 'updatedb' at 20AM every day.  Here we write the
;;  ;; job's action as a Scheme procedure.
  #~(job '(next-hour '(20))
         (lambda ()
           (execl (string-append #$findutils "/bin/updatedb")
                  "updatedb"
                  "--prunepaths=/tmp /var/tmp /gnu/store"))))

(define garbage-collector-job
;;  ;; Collect garbage 5 minutes after 17 o'clock every day.
;;  ;; The job's action is a shell command.
  #~(job "5 17 * * *"            ;Vixie cron syntax
         "guix gc -F 5G"))

(define idutils-job
;;  ;; Update the index database as user "hubert" at 12:15PM
;;  ;; and 19:15PM.  This runs from the user's home directory.
  #~(job '(next-minute-from (next-hour '(12 19)) '(15))
         (string-append #$idutils "/bin/mkid src")
```

```
          #:user "hubert"))

(operating-system
  (host-name "gnu")
  (timezone "Europe/Paris")
  (locale "fr_FR.utf8")
```

```
# guix system reconfigure /etc/config.scm
```

Voilà, les modifications ont été prises en compte.

Sources : https://www.notabug.org/jbranso/guix-config/src/master/awesome.scm
https://guix.gnu.org/manual/en/html_node/Scheduled-Job-Execution.html

Edit : Bah, j'ai oublié de tenir compte qu'il fallait aussi rajouter **service mcron-service-type** dans **%base-services**))) et peut-être même **(kernel-arguments '(“modprobe.blacklist=usbkbd”))**

Du coup je ne sais pas ce que va donner le reconfiguration

A suivre…

Ce système me donne toujours autant satisfaction malgré sa jeune existence, les quelques trucs manquants étant amplement comblés par une pratique quotidienne des plus enrichissantes. J'espère avoir le temps de donner ici d'autres exemples d'utilisation et de maintenance.