Comment refaire plus tard et là-bas ce qui a été fait aujourd'hui et ici ?

Simon Tournier

INSERM US 53 - CNRS UAR 2030
simon.tournier@u-paris.fr

15 avril 2022
https://hpc.guix.info

Le monde est *open*, n'est-ce pas ?

- *open* journal
- *open* data
- *open* source
- *open* science
- *open* etc.

Quel est le problème de reproductibilité dans un contexte scientifique ?
(même si tout devient *open*)

Exemple

Listing 1 – Fonction de Bessel

```
#include <stdio.h>
#include <math.h>

int main(){
  printf("%E\n", j0f(0x1.33d152p+1f));
}
```

Alice voit : 5.643440E-08
Carole voit : 5.963430E-08

Pourquoi ? Le code est disponible pourtant.

Établir si le différence est significative ou non est laissé à l'expertise des scientifiques du domaine.

Enjeu de la recherche reproductible

D'un point de vue la « méthode scientifique » :

Tout l'enjeu est le **contrôle de la variabilité**

D'un point de vue du « savoir scientifique » (caractère universel) :

- Un observateur indépendant doit être capable d'observer le même résultat.
- L'observation doit être pérenne (dans une certaine mesure).

Dans un monde où (presque) tout est donnée numérique (*data*),

comment refaire plus tard et là-bas ce qui a été fait aujourd'hui et ici ?

(sous entendu avec un « ordinateur »)

Nous allons discuter de...

1. Du problème d'Alice et Carole
   - Gestionnaire de paquets
   - Guix dans tout ça
2. De pérennité
   - Préservation de quoi et pourquoi ?
   - Archivage logiciel : Software Heritage
   - Guix dans tout ça
3. Ce qu'il faut retenir

(quelques exemples issus du langage `C` mais tout est transposable à n'importe quel autre pile logicielle)

Quelques questions

```c
#include <stdio.h>
#include <math.h>

int main(){
  printf("%E\n", j0f(0x1.33d152p+1f));
}
```

Alice voit : 5.643440E-08
Carole voit : 5.963430E-08

- Quel compilateur ?
- Quelles bibliothèques (`<math.h>`) ?
- Quelles versions ?
- Quelles options de compilation ?

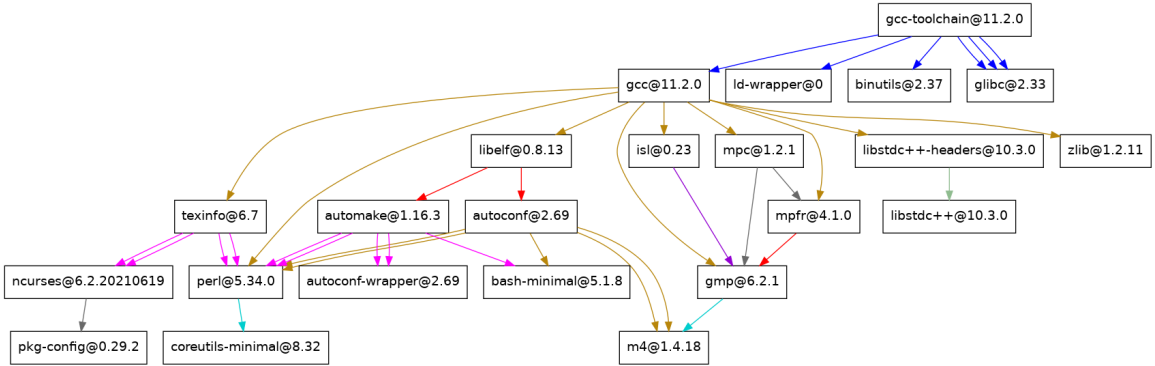
En d'autres termes

- Quelles sont les sources des outils ?
- Quelles sont les outils requis pour la construction ?
- Quelles sont les outils requis pour l'exécution ?
- Comment chaque outil est-il produit ?

Répondre à ces questions signifie **contrôler la variabilité** de l'environnement computationnel

Comment capturer ces informations ?

Alice dit « GCC à la version 11.2.0 »



Est-ce la même version de GCC si `mpfr` est à la version 4.0 ?

Graphe complet : 43 ou 104 ou 125 ou 218 nœuds
(suivant ce que l'on considère comme graine binaire du *bootstrap*)

Options de constructions (compilation)

```c
#include <stdio.h>
#include <math.h>

int main(){
  printf("%E\n", j0f(0x1.33d152p+1f));
}
```

```
alice@laptop$   gcc bessel.c                    && ./a.out
                5.643440E-08
carole@desktop$ gcc bessel.c  -lm -fno-builtin  && ./a.out
                5.963430E-08
```

(Ah, sacré *constant folding*)

Il faut capturer comment **tous** les outils (nœuds du graphe) sont produits.

Gestionnaire de paquets = gestionnaire de graphe

Comment capturer ces informations ?

- Quelles sont les sources des outils ? — `source`
- Quelles sont les outils requis pour la construction ?
- Quelles sont les outils requis pour l'exécution ?

  } `inputs`, `propagated-`, `native-inputs`
- Comment chaque outil est-il produit ? — `build-system`, `arguments`

```
(package                    ; definition du noeud python-pytorch
  (name "python-pytorch")
  (version "1.10.2")
  (source ... )
  (build-system python-build-system)
  (arguments ... )
  (inputs (list ...)))      ; liste d'autres noeuds -> graphe (DAG)
```

Révision = un graphe spécifique

version = graphe

```
$ guix describe
Generation 77 Apr 05 2022 10:27:45 (current)
  guix 20303c0
    repository URL: https://git.savannah.gnu.org/git/guix.git
    branch: master
    commit: 20303c0b1c75bc4770cdfa8b8c6b33fd6e77c168
```

La révision 20303c0 capture **tout** le graphe

- Alice dit « j'ai utilisé Guix à la révision 20303c0 »
- Carole connaît toutes les informations pour reproduire le même environnement

Paquet spécifique et canal (*channels*)

Alice développe un outil dont les sources sont à :

`https://gitlab.inria.fr/projet/un-outil.git`

et l'empaquette elle-même via un canal :

`https://gitlab.inria.fr/projet/un-canal.git`

Le dépôt Git `un-canal.git` capture les informations
pour construire l'outil à partir des sources `un-outil.git`.

Collaboration

Alice décrit son environnement :

- la révision (Guix lui-même et aussi potentiellement les autres canaux) :

```
guix describe -f channels > alice.scm
```

- la liste des paquets avec le fichier `paquets.scm`

génère son environnement avec, e.g.,

```
guix shell -m paquets.scm
```

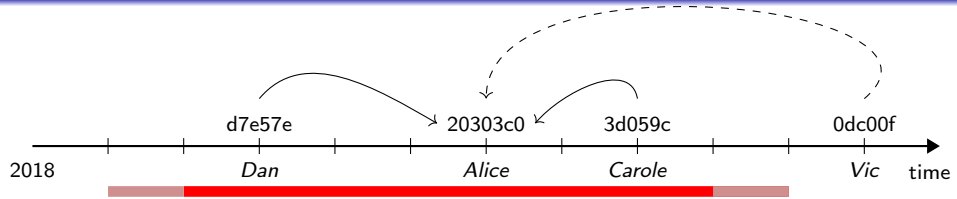
et **partage ses deux fichiers**.

Carole génère le même environnement **à partir des deux fichiers** d'Alice,

```
guix time-machine -C alice.scm -- shell -m paquets.scm
```

Dan peut donc aussi avoir le même environnement qu'Alice et Carole.

Changer de révision, en arrière, en avant : `guix time-machine`



Pour être reproductible dans le temps, il faut :

- Une préservation de **tous** les codes source
- Une *backward* compatibilité du noyau Linux
- Une compatibilité de l'*hardware* (e.g., CPU, disque dur (NVMe), etc.)

Quelle est la taille de la fenêtre temporelle avec les 3 conditions satisfaites ?

(À ma connaissance, le projet Guix réalise une expérimentation grandeur nature et quasi-unique depuis sa v1.0 en 2019.)

comment refaire plus tard et là-bas ce qui a été fait aujourd'hui et ici ?

traçabilité et transparence

être capable d'étudier bogue-à-bogue

Guix devrait s'occuper de tout

```
guix time-machine -C channels.scm -- shell -m manifest.scm
```

si on spécifie

« comment construire » `channels.scm`

« quoi construire » `manifest.scm`

comment refaire plus tard et là-bas ce qui a été fait aujourd'hui et ici ?

traçabilité et transparence

être capable d'étudier bogue-à-bogue

Guix devrait s'occuper de tout

```
guix time-machine -C channels.scm -- shell -m manifest.scm
```

si on spécifie

« comment construire » `channels.scm`

« quoi construire » `manifest.scm`

Que faut-il préserver en plus de ces 2 fichiers ?

Préservation de quoi ?

```
guix time-machine -C channels.scm -- shell -m manifest.scm
```

- chaque canal (= dépôt Git qui contient des définitions de paquet) `channels.scm`
- chaque source du paquet `manifest.scm`

```
(package                    ; definition du noeud python-pytorch
  (name "python-pytorch")
  (version "1.10.2")
  (source ... )             ; les donnees pointees ICI
  (build-system python-build-system)
  (arguments ... )
  (inputs (list ...)))
```

Préservation de quoi ? (2)

- dépôt Git (canal)
- `source`
  - archive *tarballs* (compressées) `url-fetch`
  - dépôt Git `git-fetch`
  - dépôt Subversion `svn-fetch`
  - dépôt Mercurial `hg-fetch`
  - dépôt CVS `cvs-fetch`

Préservation de quoi ? (2)

- dépôt Git (canal)
- `source`
  - archive *tarballs* (compressées) `url-fetch`
  - dépôt Git `git-fetch`
  - dépôt Subversion `svn-fetch`
  - dépôt Mercurial `hg-fetch`
  - dépôt CVS `cvs-fetch`

```
$ guix repl -- sources.scm | sort | uniq -c | sort -nr
  13432 url-fetch
   6691 git-fetch
    391 svn-fetch
     43 other
     31 hg-fetch
      3 cvs-fetch
```

Pourquoi les préserver ?

Parce que les **services en ligne parfois s'arrêtent**

- Google Code (lien) début 2016
- Alioth (Debian) en 2018 remplacé par Salsa
- Gna ! en 2017 après 13 années d'activité
- Gitourious en 2015 (le second plus populaire service d'hébergement pour Git en 2011)
- etc.

Pourquoi les préserver ?

Parce que les **services en ligne parfois s'arrêtent**

- Google Code (lien) début 2016
- Alioth (Debian) en 2018 remplacé par Salsa
- Gna ! en 2017 après 13 années d'activité
- Gitourious en 2015 (le second plus populaire service d'hébergement pour Git en 2011)
- etc.
- `gforge.inria.fr` par exemple Guix issue #42162 (lien)

```
Believe it or not, gforge.inria.fr was finally phased out on
Sept. 30th.  And believe it or not, despite all the work and all the
chat :-), we lost the source tarball of Scotch 6.1.1 for a short period
of time (I found a copy and uploaded it to berlin a couple of hours
ago).
```

Comment les préserver ?

Forge $\neq$ Archive

L'objectif d'une forge est de permettre à plusieurs développeurs de **participer ensemble au développement** d'un ou plusieurs logiciels, le plus souvent à travers le réseau Internet.

https://fr.wikipedia.org/wiki/Forge_(informatique)

L'archivage est un ensemble d'actions qui a pour but de garantir l'accessibilité sur le long terme d'informations (dossiers, documents, données) que l'on doit ou souhaite conserver pour des raisons juridiques

https://fr.wikipedia.org/wiki/Archivage

Software Heritage « *are building the universal software archive* » (lien)

Les services en ligne parfois s'arrêtent...

Pourquoi serait-il différent pour Software Heritage ?

Aucune garantie mais...

Software Heritage is an open, non-profit initiative unveiled in 2016 by Inria. It is supported by a broad panel of institutional and industry partners, in collaboration with UNESCO.

The long term goal is to collect all publicly available software in source code form together with its development history, replicate it massively to ensure its preservation, and share it with everyone who needs it.

- Supporté par des institutions
- Avec la mission d'archiver

(demo du site SWH)

État des lieux de Guix et Software Heritage (SWH)

Guix est capable de

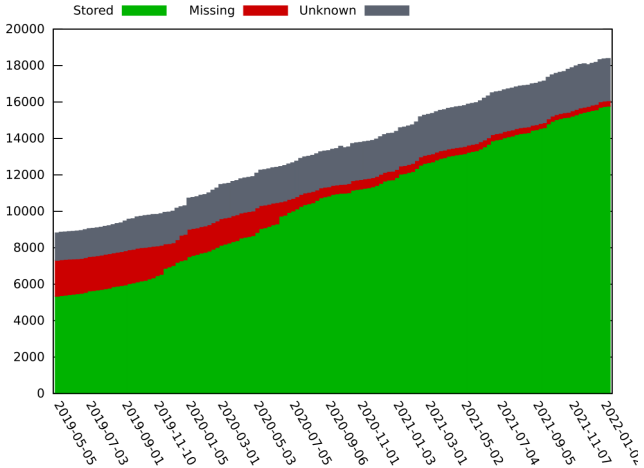
- sauvegarder (automatiquement) les sources dans SWH
- retrouver (automatiquement) les sources depuis SWH

Mais pas tous les types de sources !

- Des outils disponibles dans Guix pour les sources :
  - dépôt Git `git-fetch`
  - archives *tarballs* (compressées) `url-fetch`
- . . .et manquants pour tout le reste (aide bienvenue :-))

Quelques stats : préservation de Guix

https://ngyro.com/pog-reports/latest/



Sauvegarder dans Software Heritage

- Git = 98.3%
- *tarballs* ≈ 70%

En pratique

- Sauvegarder un paquet

```
guix lint -c archival nom-du-paquet
```

- Retrouver la source d'un paquet (si elle a « disparu »)

~~`guix help`~~ . . .bah c'est automatique :-)

Normalement, si tout se passe bien,

```
guix time-machine -C channels.scm -- shell -m manifest.scm
```

Guix va regarder, dans l'ordre,

- ce qui est disponible sur le serveur de substituts
- si les sources sont à leur endroit attendu (défini par `source`)
- s'il les trouve sur Software Heritage

Même les dépôts des canaux (`channels.scm`)

En pratique (2)

- Alice développe un outil dont les sources sont à :

  `https://gitlab.inria.fr/projet/un-outil.git`

  et l'empaquette elle-même via un canal :

  `https://gitlab.inria.fr/projet/un-canal.git`
- Alice sauvegarde sur Software Heritage ces deux dépôts
- Alice joint à sa publication les deux fichiers `channels.scm` et `manifest.scm`
- Carole génère le même environnement qu'Alice avec :

  ...même si les serveurs `inria.fr` sont éteints !

```
guix time-machine -C channels.scm -- shell -m manifest.scm
```

En pratique (3)

```
$ guix time-machine -C channels.scm -- shell -m manifest.scm
Updating channel 'guix' from Git repository at 'https://git.savannah.gnu.org/gi
Updating channel 'example' from Git repository at 'https://whatever-here.org/do
SWH: found revision 67c9f2143aa6f545419ae913b4ae02af4cd3effc with directory at
SWH vault: requested bundle cooking, waiting for completion...
swh:1:rev:67c9f2143aa6f545419ae913b4ae02af4cd3effc.git/
[...]
fatal: could not read Username for 'https://github.com': No such device or addr
Trying content-addressed mirror at berlin.guix.gnu.org...
Trying to download from Software Heritage...
SWH: found revision e1eefd033b8a2c4c81babc6fde08ebb116c6abb8 with directory at'
[...]
```

https://simon.tournier.info/posts/2021-10-25-software-heritage.html

Oui, mais. . .

- Comment identifier mon code ?

  Identifiant intrinsèque vs extrinsèque (lien)

  tag (v1.2.3) vs hash (20303c0), quel hash ? etc.
- Comment s'assurer que tout le graphe est bien préservé ?
- Que se passe-t-il s'il manque les sources d'un seul nœud ?
- On n'a pas parlé du *boostrap* (les racines du graphes).
- etc.

En bref

- Une commande

```
guix time-machine -C channels.scm -- shell -m manifest.scm
```

  et deux fichiers
- Si c'est dans Guix, des efforts (collectif et infrastructure) pour la préservation
- Si c'est dans un canal séparé, ça fonctionne mais plus de boulot « manuel »
- Difficile de garantir 100% car il y a beaucoup de cas particuliers
- Des progrès. . .et d'autres à venir ?

À vous de compléter

Des questions ?

guix-science@gnu.org

https://hpc.guix.info/events/2022/café-guix/