Outils logiciels de développement

Vincent Danjean

12 avril 2019

Sommaire

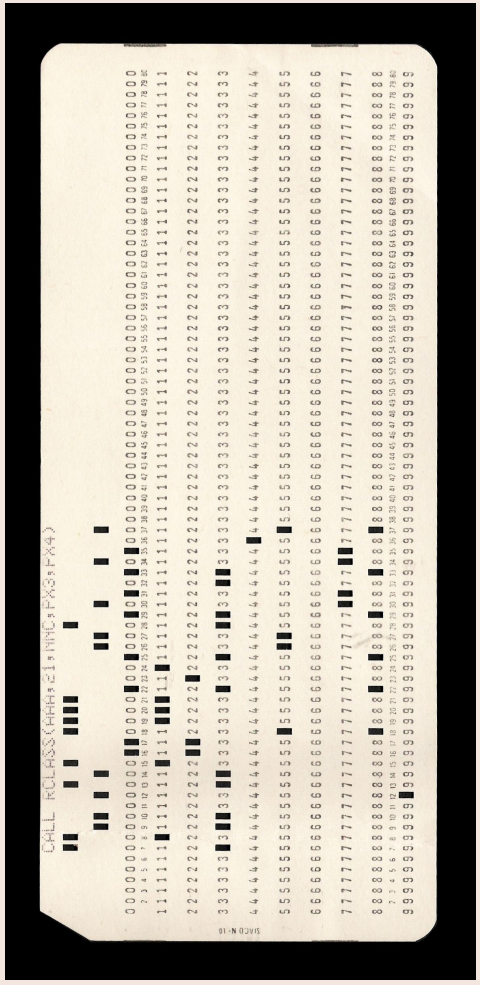
1. Introduction
2. Beaucoup d'outils pour de nombreux usage
3. Licences
4. Logiciels de contrôle de version

Au temps des dinosaures

Programmation par cartes perforées

- écriture du programme en perforant les cartes
- dépôt pour compilation et exécution dans l'ordinateur
- récupération du listing de résultat le lendemain
  - par exemple : "missing ' ;' at line 2"

Carte perforée



Au temps des dinosaures

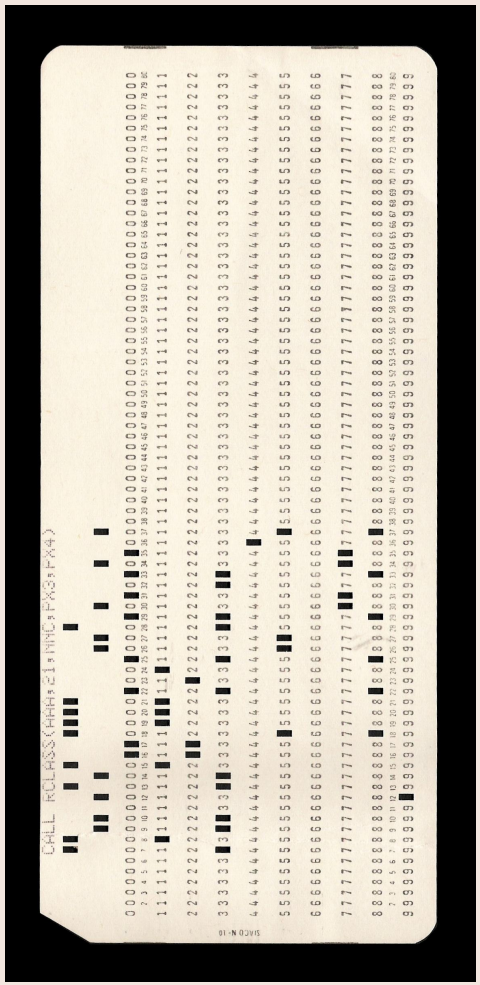
Programmation par cartes perforées

- écriture du programme en perforant les cartes
- dépôt pour compilation et exécution dans l'ordinateur
- récupération du listing de résultat le lendemain
  - par exemple : "missing ' ;' at line 2"

On trouve encore des restes de cette époque :

- terminaux de 80 colonnes par défaut
- encodage des caractères (ASCII, BCD, etc.)

Carte perforée



Des outils pour quoi faire ?

Des outils pour quoi faire ?

- concevoir
- écrire les lignes de code
- documenter
- génération de l'exécutable
- tester
- déboguer
- distribuer
- collaborer
- exécuter
- et bien d'autres choses

Des outils pour quoi faire ?

- concevoir
- écrire les lignes de code
- documenter
- génération de l'exécutable
- tester
- déboguer
- distribuer : focus sur les licences
- collaborer : focus sur les logiciels de contrôle de version
- exécuter
- et bien d'autres choses

Encore et toujours de nouveaux outils

Le développement d'outils est dû

- aux capacités du matériel
  - temps des programmes d'analyse
  - utilisation des nouvelles fonctionnalités matériels
    - compteurs de performance des processeurs
- aux nouveaux concepts qui voient le jour
  - nouveaux langages
  - méthodes de tests
- aux nouvelles organisations de travail
  - développement collaboratif
  - cycles de développement cours
    - méthodes agiles, scrum, etc.

Sommaire

1. Introduction
2. Beaucoup d'outils pour de nombreux usage
3. Licences
4. Logiciels de contrôle de version

Outils pour l'analyse et la conception

Quels objectifs ?

- établir des spécifications
  - décrire les fonctionnalités
  - répondre à la question "Quoi ?"
- concevoir
  - décrire les moyens et fonctions pour obtenir les fonctionnalités
  - répondre à la question "Comment ?"

Exemple d'outils

- UML
- Merise
- etc.

Outils probablement peu/pas utilisés pendant cette formation (sauf papier/crayon)

Outils pour l'écriture des programmes

Quels objectifs ?

écrire du code :

- éditeur dédié au code :
  - coloration syntaxique, indentation, folding, auto-complétion, refactorisation, etc.
- génération automatique de code :
  - à partir du dessin d'une interface
  - à partir de spécifications formelles (gramaires, etc.)
  - etc.

Exemple d'outils

- emacs / vim
- gedit
- eclipse (+PyDev)
- Sublime Text
- Atom/Atom-IDE
- etc.
- Lex/Yacc, JavaCC, Grako

Il est essentiel de se trouver un bon editeur

mais ne pas hésiter à en essayer plusieurs

Outils pour générer de la documentation

Quels objectifs ?

documentation utilisateur

- formats facile à éditer (texte)
- compilation vers différents formats (PDF, HTML, etc.)

documentation développeur

- extraction (partielle) depuis le code

Exemple d'outils

formats textes :

- markdown, org-mode, LaTeX

Compilateurs textes :

- pandoc, emacs, pdfLaTeX, etc.

Extraction de documentation :

- doxygen, javadoc, pydoc, pyreverse

Documenter son code est essentiel

penser à l'autodocumentation (noms de fonctions, de paramètres, de variables)

Outils pour la production des exécutables

Quels objectifs ?

générer un exécutable

- par transformation
  - compilation
- par une succession d'étapes
  - système de construction

Exemple d'outils

Compilateurs :

- gcc, clang, cython

Systèmes de constructions :

- Make, Autotools, CMake, Maven, distutils, etc.

Python est généralement interprété (pas de compilation)
mais on peut construire des paquets à distribuer

Outils pour tester les programmes

Quels objectifs ?

Tester des fonctions, des comportements

- boîte blanche/noire/grise
- tests unitaires, d'intégration, de non-régression

Tester des aspects non-fonctionnels

- passage à l'échelle, etc.

Automatisation des tests :

- scripts
- intégration continue

Exemple d'outils

- bash, XUnit (PyUnit), autotest, CMake, etc.
- jenkins, gitlab, etc.

Un bon programmeur écrit des tests avant son programme

Le test est une partie cruciale du développement
Le test ne se fait pas qu'après l'écriture de tout le code

Outils pour la mise au point des programmes

Quels objectifs ?

Exécuter un programme pas à pas

Observer l'exécution d'un programme

Analyser :

- le code source d'un programme
- l'exécution d'un programme

Exemple d'outils

déboguer :

- gdb, lldb, pdb, etc.

analyser :

- valgrind, pylint, etc.
- pythontutor.com

Déboguer est une tâche difficile...

...mais essentielle. L'enseignement de ces outils est très difficile.

- démarche à acquérir (nécessite de se remettre en cause)
- savoir douter de tout
- prendre le temps de maîtriser les outils disponibles

Le premier bug

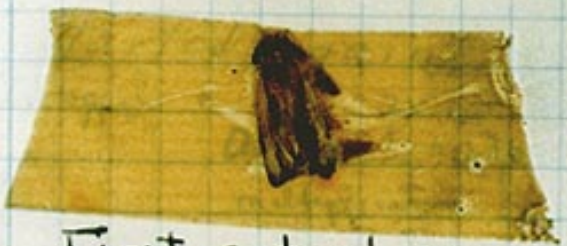
Ce bug a été identifié en 1947 et pris en photo...

Le premier bug

Ce bug a été identifié en 1947 et pris en photo...

9/9

0800 Arctan started
1000 " stopped - arctan ✓
13"u/c (032) MP - MC 4.615925059(-2)
(033) PRO 2 2.13047 6415
correct 2.13067 6415
Relays 6-2 in 033 failed special speed test
In relay " " test.
Relays changed
1100 Started Cosine Tape (Sine check)
1525 Started Mult + Adder Test.



1545 Relay #70 Panel F
(moth) in relay.
First actual case of bug being found.
1700 closed down.

Outils pour distribuer le code

Quels objectifs ?

Gérer les dépendances

- peuvent être récursives

Gérer les bibliothèques

- récupération et installation

Exemple d'outils

- pip, Conda, CPAN, CTAN, CRAN, etc.

Ces outils sont souvent associés à des sites web de dépôt

Outils pour le travail collaboratif

Quels objectifs ?

Partager du code et des idées

- forums, archives, listes de diffusions (mail), etc.
- attention aux licences !

Développer du code à plusieurs

- logiciels de contrôle de version

Exemple d'outils

- stackoverflow, github, gitlab, etc.
- Git, subversion, mercurial, etc.

Les licences et les logiciels de contrôle de version seront approfondis par la suite

Outils pour exécuter les programmes

Quels objectifs ?

Garantir un environnement stable/reproductible pour l'exécution d'un programme :

- virtualisation
  - la machine (virtuelle) entière est figée
- containerisation
  - l'environnement (plus ou moins complet) est figé

Exemple d'outils

virtualisation :

- kvm, virtualBox, etc.

containerisation :

- docker, singularity
- virtualenv

Sommaire

1. Introduction
2. Beaucoup d'outils pour de nombreux usage
3. Licences
4. Logiciels de contrôle de version

Quelques considérations légales

Je ne suis pas juriste

En cas de doutes et de besoin de certitudes, consulter un juriste

Les logiciels et les documents produits relèvent du droit d'auteur

pas (ou très peu) de droits par défaut

- sans licence et sans être auteur, on ne peut rien faire

attention au plagiat (faire passer pour sien un travail fait par d'autres)

Licences libres : 4 libertés indissociables

1. la possibilité d'utiliser l'œuvre, pour tous les usages
2. la possibilité d'étudier l'œuvre
3. la possibilité de redistribuer des copies de l'œuvre
4. la possibilité de modifier l'œuvre et de publier ses modifications

Que faire en pratique

Conseils de bonnes pratiques

Mettre une licence sur tous les documents qu'on écrit

- réfléchir à l'usage qu'on souhaite en permettre

Ne pas enlever les anciens auteurs quand on reprend/améliore un document

Citer ses sources

Licences classiques

	code	documents
Je donne le source, on peut en faire ce qu'on en veut	BSD	CC-BY
Je donne le source, les documents dérivés doivent rester libre	GPLv3/GPLv2+	CC-BY-SA
J'autorise juste la redistribution du résultat final	propriétaire	CC-BY-NC et/ou ND
Je veux vendre de manière exclusive mon résultat final	propriétaire	propriétaire/ éditeur

Sommaire

1. Introduction
2. Beaucoup d'outils pour de nombreux usage
3. Licences
4. Logiciels de contrôle de version