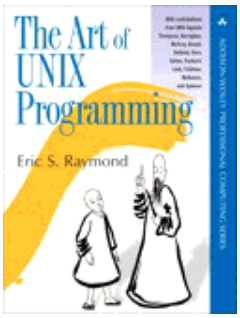
LinuxFocus article number 313
http://linuxfocus.org

Lecture: L’Art de la programmation UNIX



par Edgar Hernández Zúñiga
<edgar(en)linuxfocus.org>

L´auteur:

Je n’ai pas de biographie, même pas une petite biographie ...

Traduit en Français par:
Guy Passemard
<g.passemard(at)free.fr>

Résumé:

Cet article va essayer de se focaliser sur les principaux sujets présentés dans ce livre. Lorsque vous lirez ce texte, le livre devrait être disponible dans les magasins. L’article est basé sur la version 0.87 du livre, c’est une version bêta qui nous a été fourni pour faire une évaluation avant sa parution. En écrivant cet article, j’ai réalisé toute l’importance des sujets abordés. "L’art de la programmation UNIX" pourrait mériter un article à lui seul. Le livre est trés bien écrit et vous vous apercevrez qu’Eric sait de quoi il parle.

titre révisé:	L’Art de la Programmation UNIX.
Auteur(s):	Eric S. Raymond.
Contributions:	Thompson, Kernighan, McIlroy, Arnold, Bellovin, Korn, Gettys, Packard, Lesk, Feldman, McKusick, Spencer.
Pages:	550 dans cette version.
Editeur:	Addison Wesley (http://www.awprofessional.com)

Introduction

Eric S. Raymond connu pour son article "The Cathedral and the Bazaar", a écrit un livre exceptionnel. Ce livre nous introduit dans l’univers technologique et les éléments du systèmes Unix.

Le travail de Eric S. Raymond est soutenu par d’autres grands noms du monde UNIX qui ont contribué à ce livre. Parmi eux Ken Thompson, Brian Kernighan and Dennis Ritchie.

Le livre est divisé en quatre sections principales :

- Contexte
- Conception
- Implémentation/Outils
- Communauté

Dans chacune de ces sections les sujets sont répartis sur des thèmes allant des *Bases de la Philosophie Unix*, aux *Bonnes Pratiques pour travailler avec les Développeurs "Open-Source"* puis sur les concepts de *La Modularité, La Conception de protocols pour les applications, la Transparence, les Mini-langages and la Complexité*. Ensuite les *Langages et Outils* décrivant l'implémentation, et en plus de tout ça, vous trouverez des exemples pratiques qui permettront aux lecteurs de mieux comprendre.

Ci-dessous est présenté la table des matières du livre. Elle décrit brièvement chaque point abordé, et cela peut être utile pour se faire une idée des sujets développés dans ce livre.

Table des matières

Unix and object-oriented languages.
Coding for modularity.

5. Textuality.

The Importance of Being Textual.
Data file meta-formats.
Application protocol design.
Application protocol meta-formats.

6. Transparency.

Some case studies.
Designing for transparency and discoverability.
Designing for maintainability.

7. Multiprogramming.

Separating complexity control from performance tuning.
Taxonomy of Unix IPC methods.
Problems and methods to avoid.
Process partitioning at the design level.

8. Minilanguages.

Taxonomy of languages.
Applying mini-languages.
Designing mini-languages.

9. Transformation.

Data-driven programming.
Ad-hoc code generation.

10. Configuration.

What should be configurable?
Where configurations live.
Run-control files.
Environment variables.
Command-line options.
How to choose among configuration-setting methods.
On breaking these rules.

11. Interfaces.

Applying the Rule of Least Surprise.
History of interface design on Unix.
Evaluating interface designs.
Tradeoffs between CLI and visual interfaces.
Transparency, expressiveness, and configurability.
Unix interface design patterns.
Applying Unix interface-design patterns.
The Web browser as universal front end.
Silence is golden.

12. Optimization.

Don’t just do something, stand there!
Measure before optimizing.
Non-locality considered harmful.

Throughput vs. latency.

13. Complexity.

Speaking of complexity.
A Tale of Five Editors.
The right size for an editor.
The right size of software.

III. IMPLEMENTATION.

14. Languages.

Unix's Cornucopia of Languages.
Why Not C?
Interpreted Languages and Mixed Strategies.
Language evaluations.
Trends for the Future.
Choosing an X toolkit.

15. Tools.

A developer-friendly operating system.
Choosing an editor.
Special-purpose code generators.
Make in non-C/C++ Development.
Version-control systems.
Run-time debugging.
Profiling.
Emacs as the universal front end.

16. Re-Use.

The tale of J. Random Newbie.
Transparency as the key to re-use.
From re-use to open source.
The best things in life are open.
Where should I look?
What are the issues in using open-source software?
Licensing issues.

IV. COMMUNITY.

17. Portability.

Evolution of C.
Unix standards.
Specifications as DNA, code as RNA.
Programming for Portability.
Internationalization.
Portability, open standards and open source.

18. Documentation.

Documentation concepts.
The Unix style.
The zoo of Unix documentation formats.
The present chaos and a possible way out.
The DocBook tool chain.
How to write Unix documentation.

19. Open Source.

```
    Unix and open source.
    Best practices for working with open-source developers.
    The logic of licenses: how to pick one.
    Why you should use a standard license.
    Varieties of Open-Source Licensing.
```

20. Futures.

```
    Essence and accident in Unix tradition.
    Problems in the design of Unix.
    Problems in the environment of Unix.
    Problems in the culture of Unix.
    Reasons to believe.
```

A. Glossary of Abbreviations.

B. References.

C. Contributors.

La Culture et la Philosophie Unix

Pour ceux qui ont une expérience dans le monde Unix,ou dans un autre système d'exploitation dérivé d'Unix il n'est pas rare d'entendre le terme : *Philosophie*, Unix c'est une philosophie, en plus d'une culture c'est une façon de vivre, une façon de faire les choses, une façon de concevoir et de programmer.

Unix est basé sur une puissante philosophie et une conception robuste qui lui ont permis depuis sa création en 1969 d'être la référence pour la réalisation de nombreux systèmes d'exploitation.

Les Bases de la Philosophie Unix

La philosophie Unix a commencé évidemment avec Ken Thompson lorsqu'il voulut concevoir un système d'exploitation, petit mais efficace . Depuis beaucoup de personnes ont contribuées à son évolution et Unix aujourd'hui représente l'expérience venant de divers horizons.

Le livre contient de nombreux sujets de grand interêt à étudier. Une partie que j'ai trouvé particulièrement utile est celle qui encourage le lecteur d'extraire l'essentiel lors de la conception avant de commencer la réalisation et de réfléchir aux principes de base contenus dans la philosophie Unix.

- *La Modularité:* Ecrire des modules simples reliés par des interfaces claires.
- *La Composition:* Concevoir des programmes qui pourront être interfacés avec d'autres.
- *L'Economie:* Le temps du programmeur coûte cher: préservez-le et consommez plutôt du temps

machine.

- *L’Optimization:* Réaliser le prototype avant de peaufiner. Faite le marcher avant de l’optimiser.
- *L’Extensibilité:* Concevoir pour le futur, car il arrivera plus tôt que vous le pensez

Conclusion et Recommandations

C’est, je l’affirme, un excellent livre. Eric Steven Raymond vous apprend comment écrire du bon logiciel et avant de commencer, de penser aux concepts Unix qui guide votre propre conception. Le livre devrait être aussi trés intéressant pour tous les programmeurs qui savent écrire du logiciel en C, C++ Java etc ... mais qui arrivent d’autres systèmes d’exploitation.

La bibliographie est extensive et permet d’élargir vos horizons. J’ai personnellement apprécié les références à la documentation sur les systèmes de contrôle de versions.

J’espère avoir écrit un article qui d’une certaine façon présente une analyse plus approfondie du livre, tout en souhaitant que vous avez apprécié cette brève introduction.

Site Web maintenu par l´équipe d´édition LinuxFocus	Translation information:
© Edgar Hernández Zúñiga	es --> -- : Edgar Hernández Zúñiga <edgar(en)linuxfocus.org>
"some rights reserved" see linuxfocus.org/license/	es --> en: Edgar Hernández Zúñiga <edgar(en)linuxfocus.org>
http://www.LinuxFocus.org	

2005-01-14, generated by lfparser_pdf version 2.51