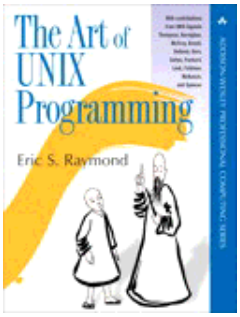
LinuxFocus article number 313
http://linuxfocus.org

Boekbespreking: The Art of UNIX Programming



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

Over de auteur:

Ik heb geen biografie, zelfs geen kleine...

Vertaald naar het Nederlands door:
Guus Snijders
<ghs%28at%29linuxfocus.org>

Kort:

Deze bibliografische bespreking probeert een duidelijk en beknopt perspectief te geven van de onderwerpen die in dit boek zijn opgenomen dat waarschijnlijk in de winkel ligt op het moment dat je dit artikel leest.

Deze bespreking is gebaseerd op versie 0.87 van het boek, een tekst die we ontvingen ter evaluatie, juist voordat het gepubliceerd werd.

Terwijl ik dit artikel aan het schrijven was, realiseerde ik me dat het onderwerp van het boek zo breed was, en de manier waarop het gepresenteerd wordt zo waardevol dat we wellicht een ander artikel zouden moeten opnemen om een diepere evaluatie te geven van de onderwerpen van dit boek.

Besproken titel:	The Art of UNIX Programming.
Auteur(s):	Eric S. Raymond.
Bijdragen:	Thompson, Kernighan, McIlroy, Arnold, Bellovin, Korn, Gettys, Packard, Lesk, Feldman, McKusick, Spencer.
Pagina’s:	550 in deze versie.
Uitgever:	Addison Wesley (http://www.awprofessional.com)

Introductie

Eric S. Raymond, bekend van de Cathedral and the Bazaar, heeft geweldig werk geleverd. Dit boek zal binnenkort gepubliceerd worden, maar het is ook mogelijk om het online te vinden. In feite geeft deze geweldige compilatie, gebaseerd op vele uren van research en processen, ons een algemeen idee van de vele verschillende technologieën en elementen die gerelateerd zijn aan Unix systemen, als ook sommige van hun resultaten.

Het werk van Eric S. Raymonds wordt ondersteund door vele co-werkers zoals Ken Thompson, Brian Kernighan en Dennis Ritchie. Opmerkelijk is dat de auteur toegeeft dat de eerste en de laatste hem ertoe aangezet hebben dit werk te doen.

Het boek is verdeeld in 4 delen:

- Context
- Design
- Implementation/Tools
- Community

Ieder deel bevat verschillende onderwerpen, van Basics of the Unix philosophy, in het hoofdstuk Context tot Best Practices for Working with Open-Source developers, alsook concepten van Modularity, Design protocols for Applications, Transparency, Minilanguages and Complexity in het hoofdstuk Design, en Languages and Tools onder Implementation. Verder zijn illustratieve, praktische voorbeelden opgenomen wanneer die nodig waren om de lezer een beter beeld te geven.

Tot mijn spijt, daar ik de voorkeur geef aan duidelijke besprekingen van boeken boven het noemen van de inhoud en wat korte commentaren, besloot ik om de index van dit boek op te nemen. Ik ging er van uit dat de lezers het bruikbaar zouden vinden.

Inhoudsopgave

Origins and history of Unix, 1969-1995.
Origins and history of the hackers, 1961-1995.
The open-source movement: 1998 and onward.
The lessons of Unix history.

3. Contrasts.
The elements of operating-system style.
Operating-system comparisons.
What goes around, comes around.

II. DESIGN.

4. Modularity.
Encapsulation and optimal module size.
Compactness and orthogonality.
Libraries.
Unix and object-oriented languages.
Coding for modularity.

5. Textuality.
The Importance of Being Textual.
Data file metaformats.
Application protocol design.
Application protocol metaformats.

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 minilanguages.
Designing minilanguages.

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.

Unix cultuur en filosofie

Zoals ik eerder al noemde, behalve het geven van een persoonlijke mening over dit boek, denk ik dat het

ook verstandig is om een systematische analyse te overwegen die ons niet alleen de inhoud van het boek oplevert, maar ook een duidelijk idee voor die lezers die niet in staat zijn om alles te lezen.

Voor de lezers die al door de Unix wereld zijn gegaan en ook sommige gerelateerde besturingssystemen, zal de uitdrukking *Philosophy* niet onbekend zijn. Unix is niet alleen een filosofie, maar een cultuur. Het is bijvoorbeeld, een benadering van een compleet andere manier van programmeren.

Unix is gebaseerd op een krachtige filosofie van ontwerp welke, sinds het begin in 1969, een referentie vormt voor het creëren van toekomstige besturingssytemen.

Basis van de Unix Filosofie

De Unix filosofie, begonnen door Ken Thompson, die geïnteresseerd was in het creëren van een simpel doch competent besturingssysteem, en de geleerde lessen, geleverd door verschillende bronnen kunnen niet beschouwd worden als een formele ontwerpmethode maar een filosofie gebaseerd op ervaring.

Dit boek levert de lezer geweldige onderwerpen om te bestuderen. Een van de meest gedurfde delen is dat waar we worden aangemoedigd om onze gedachten links te laten liggen en de volgende basisideeën te onthouden, welke onderdeel zijn van de Unix filosofie:

- *Modulariteit:* Schrijf eerst eenvoudige delen, verbonden door schone interfaces.
- *Compositie:* Ontwerp programma's om verbonden te worden met andere programma's.
- *Economy:* Programmatijd is duur: bespaar bij voorkeur op machine tijd.
- *Optimalisatie:* Prototype alvorens af te werken. Zorg dat het werkt voordat je optimaliseert.
- *Uitbreidbaarheid:* Ontwerp voor de toekomst, omdat die hier sneller kan zijn dan je denkt.

Conclusie en aanbevelingen

Zonder twijfel is dit een geweldig boek. Eric Steven Raymond heeft een geweldige compilatie samengesteld in een erg goede stijl toegevoegd aan de bevestigde referenties, zoals Unix Programming Environment van Kernighan en Pike, Unix Philosophy door Gancarz en The Pragmatic Programmer van Hunt en Thomas, onder anderen. Dit alles maakt lezen een noodzaak.

Het boek probeert om die programmeurs die nog beginners zijn de nodige ervaring op te doen om zichzelf te verbeteren, maar zal ook van nut zijn voor programmeurs die niet bekend zijn met Unix maar een redelijke kennis hebben van talen als C, C++ en Java.

Het boek bevat een uitgebreide bibliografie voor die lezers die geïnteresseerd zijn in meer vaardigheden. Ik persoonlijk genoot van het lezen over het onderwerp over standaarden voor het samenstellen van

documentatie en versie controle systemen.

Hopelijk zal ik in staat zijn een artikel te schrijven dat ons in staat stelt veel meer te weten te komen over dit boek. Voor nu hoop ik dat je plezier hebt beleefd aan deze korte introductie.

2005-01-14, generated by lfparser_pdf version 2.51