

開始日	2006/12/31
終了日	2006/12/31

まとめ

乱暴に言ってしまうと、8-9ページにある9箇条のUNIX哲学（下記参照）について、その詳しい意味を軽妙な語り口で伝えているだけの本。

でも、先人の考え方を学べるから読む価値のある本。読んだ価値は十分にあったかな。

- スモール・イズ・ビューティフル (Small is beautiful)
- 一つのプログラムには一つのことをうまくやらせる (Make each program do one thing well)
- できるだけ早く試作する (Build a prototype as soon as possible)
- 効率より移植性を優先する (Choose portability over efficiency)
- 数値データはASCIIフラットファイルに保存する (Store numerical data in flat ASCII files)
- ソフトウェアを槌子（てこ）として使う (Use software leverage to your advance)
- シェルスクリプトによって槌子の効果と移植性を高める (Use shell scripts to increase leverage and portability)
- 過度の対話的インタフェースを避ける (Avoid captive user interfaces)
- すべてのプログラムをフィルタとして設計する (Make every program a filter)

感想

UNIXが当初からどのようなイメージの元に作成されているか、そしてUNIX的なプログラムとはどうあるべきかを学ぶことができる、他に無い良書でした。

実際、プログラム言語の書法などについては過去から何百冊も出版されていますが、UNIX上で作るとしたらどうあるべきかについて触れられているのはあまり見たことがありません。

この本で言う「UNIX的」という言葉を僕なりに解釈すると

- 大きなツールを都度用意するのではなく、小さなツールを組み合わせることで目的を達成する
- 移植性があること
- 他者の優れたプログラムを最大限活用・応用して目的を達成する

だと思います。

新しいことを発明した人が優れているのと同様、先人の資産に味付けをすることでより優れた資産を作り上げる人をも賞賛する文化です。

そしてこれらの哲学を実践することで、

- どのような状態にも柔軟に対応できるシステムができる
- 長生きするプログラム・システム（UNIX自身を含む）ができる
- コンピュータを可能な限り働かせられる（人間の思考中などによるボトルネックの解消）

というメリットを享受できる、というように受け取りました。

この考え方の一部は、UNIX上でプログラミングをするしなないに関わらず有効であると思います。たとえばスモール・イズ・ビューティフルでは、オブジェクト指向でも構造化設計でも避けるべき「何でも屋オブジェクト/モジュール」を敵と見なしています。また、「できるだけ早く試作する」という点では、現在のLightWeight Language開発で盛んに言われているラピッドプログラミング・ダックタイピングの考え方に似ています。

原著は1995年と古く、またページ数わずか148ページの読み物ですが、一度は読んでおくべき本ですね。先人の残した結果ではなく「考えていたこと・考えてほしいこと」を学べますので、

ちなみにもっと詳しいレビューは<http://www.linux.or.jp>の中でも紹介されていますので、そちらも見たほうが良いと思います。

僕は上のサイトを知らず本屋でたまたま目にして衝動買いしたのですが、買ってよかった。

ただ、組み込みLinuxという視点を持ったときには、本書で掲げられている全哲学の徹底は難しいなと感じます。組み込みLinuxではメモリ量が少ないために、プロセスを沢山起動することが許されません（プロセス起動のたびにスタックサイズががっばり取られてしまうため）。スピードのためには移植性を犠牲にする必要があったりしますので、これも哲学に反します。

なので例えばプログラムではなく内部のモジュール・タスクに哲学を適用するというように、柔軟に見る必要がありますね。