The Art of UNIX Programming

～UNIXという考え方～

はじめに

- Unix的思想は禅問答の様に普遍的な経験則の塊である。

- コンピュータシステムだけに限らず、あらゆる事象をより良くするための助けになる。

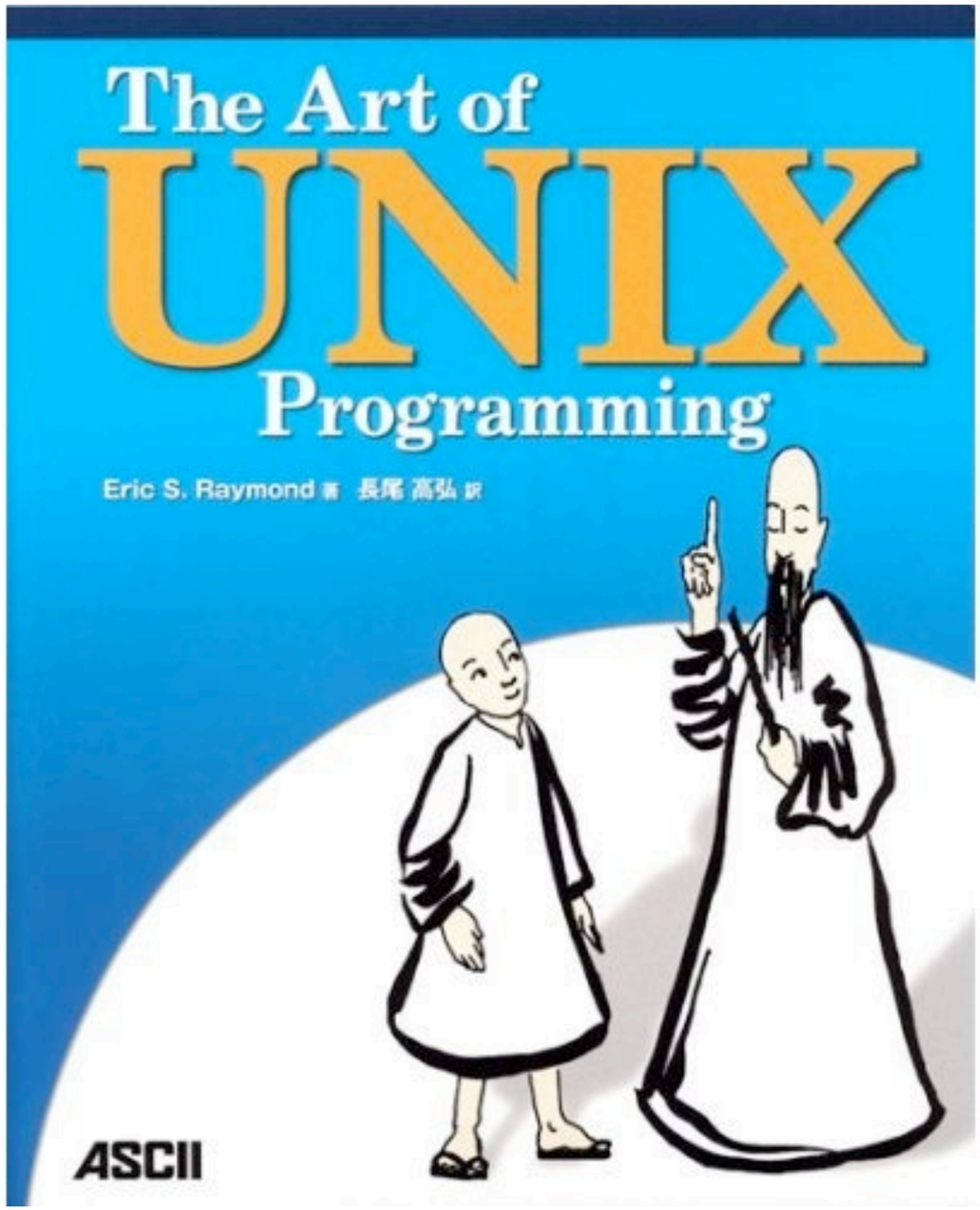
- 何の脈絡もない育児とUnixという2つの事柄を例示してその「普遍性」が伝わったなら…

まずはUNIXという考え方のおさらいを

参考図書

と言うか、これを読んで閃きました。

The Art of UNIX Programming



- **Eric S.Raymond** (著), 長尾 高弘 (翻訳)
- 大型本: 560ページ
- 出版社: アスキー (2007/6/19)
- 言語 日本語
- **ISBN-10:** 4756149480
- **ISBN-13:** 978-4756149480
- 発売日： 2007/6/19
- 商品の寸法: 23.4 x 18.6 x 3.2 cm
- ￥ **6,090**

導師からの口伝

- UNIXの設計思想は先人達の知識と現実的な経験によって作られている。

- その思想は、経験に基づくモノであるがゆえに、UNIX というOSを超えて、普遍的に通用するものである。

導師からの口伝

- UNIXの設計思想は先人達の知識と現実的な経験によって作られている。

- その思想は、経験に基づくモノであるがゆえに、UNIX というOSを超えて、普遍的に通用するものである。
- UNIXとは哲学である。

導師からの口伝

1. モジュール化の原則
2. 明確性の原則
3. 組み立て部品の原則
4. 分離の原則
5. 単純性の原則
6. 倹約の原則
7. 透明性の原則
8. 安定性の原則
9. 表現性の原則
10. 驚き最小の原則
11. 沈黙の原則
12. 修復の原則
13. 経済性の原則
14. 生成の原則
15. 最適化の原則
16. 多様性の原則
17. 拡張性の原則

導師からの口伝

1. モジュール化の原則
2. 明確性の原則
3. 組み立て部品の原則
4. 分離の原則
5. 単純性の原則
6. 倹約の原則
7. 透明性の原則
8. 安定性の原則
9. 表現性の原則
10. 驚き最小の原則
11. 沈黙の原則
12. 修復の原則
13. 経済性の原則
14. 生成の原則
15. 最適化の原則
16. 多様性の原則
17. 拡張性の原則

導師からの口伝

1. モジュール化の原則
2. 明確性の原則
3. 組み立て部品の原則
4. 分離の原則
5. 単純性の原則
6. 倹約の原則
7. 透明性の原則
8. 安定性の原則
9. 表現性の原則
10. 驚き最小の原則
11. 沈黙の原則
12. 修復の原則
13. 経済性の原則
14. 生成の原則
15. 最適化の原則
16. 多様性の原則
17. 拡張性の原則

簡潔に・単純に

導師からの口伝

1. モジュール化の原則
2. 明確性の原則
3. 組み立て部品の原則
4. 分離の原則
5. 単純性の原則
6. 倹約の原則
7. 透明性の原則
8. 安定性の原則
9. 表現性の原則
10. 驚き最小の原則
11. 沈黙の原則
12. 修復の原則
13. 経済性の原則
14. 生成の原則
15. 最適化の原則
16. 多様性の原則
17. 拡張性の原則

独立性と接続性

導師からの口伝

1. モジュール化の原則
2. 明確性の原則
3. 組み立て部品の原則
4. 分離の原則
5. 単純性の原則
6. 倹約の原則
7. 透明性の原則
8. 安定性の原則
9. 表現性の原則
10. 驚き最小の原則
11. 沈黙の原則
12. 修復の原則
13. 経済性の原則
14. 生成の原則
15. 最適化の原則
16. 多様性の原則
17. 拡張性の原則

コード生成

導師からの口伝

UI設計

1. モジュール化の原則
2. 明確性の原則
3. 組み立て部品の原則
4. 分離の原則
5. 単純性の原則
6. 倹約の原則
7. 透明性の原則
8. 安定性の原則
9. 表現性の原則
10. 驚き最小の原則
11. 沈黙の原則
12. 修復の原則
13. 経済性の原則
14. 生成の原則
15. 最適化の原則
16. 多様性の原則
17. 拡張性の原則

導師からの口伝

1. モジュール化の原則
2. 明確性の原則
3. 組み立て部品の原則
4. 分離の原則
5. 単純性の原則
6. 倹約の原則
7. 透明性の原則
8. 安定性の原則
9. 表現性の原則
10. 驚き最小の原則
11. 沈黙の原則
12. 修復の原則
13. 経済性の原則
14. 生成の原則
15. 最適化の原則
16. 多様性の原則
17. 拡張性の原則

まず動かせ

導師からの口伝

1. モジュール化の原則
2. 明確性の原則
3. 組み立て部品の原則
4. 分離の原則
5. 単純性の原則
6. 倹約の原則
7. 透明性の原則
8. 安定性の原則
9. 表現性の原則
10. 驚き最小の原則
11. 沈黙の原則
12. 修復の原則
13. 経済性の原則
14. 生成の原則
15. 最適化の原則
16. 多様性の原則
17. 拡張性の原則

未知への対応

導師からの口伝

1. モジュール化の原則
2. 明確性の原則
3. 組み立て部品の原則
4. 分離の原則
5. 単純性の原則
6. 倹約の原則
7. 透明性の原則
8. 安定性の原則
9. 表現性の原則
10. 驚き最小の原則
11. 沈黙の原則
12. 修復の原則
13. 経済性の原則
14. 生成の原則
15. 最適化の原則
16. 多様性の原則
17. 拡張性の原則

乳幼児の実装モデルと
UNIXの哲学

～ここから本題～

簡潔に・単純に

独立性と接続性

コード生成

UI設計

まず動かせ

未知への対応

簡潔に・単純に

独立性と接続性

まず動かせ

未知への対応

コード生成

UI設計

まず動かせ

まず動かせ

概ね42週でリリース。

全身の各機能は全てβ状態でリリースされる。

神経、筋肉は各個にデバッグ→機能強化される。

※ゼネラルムーブメント、原始反射等(本人の意思で身体を動かせる状態ではない。)

簡潔に・単純に

簡潔に・単純に

各機能はそれぞれがシンプルな機能だけを持ってリリースされる。

指を曲げる、物を見る(ピントの調節機能は未実装)、手足を動かす(手足は連動して全部動く)。

独立性と接続性

独立性と接続性

各機能はそれぞれ独立した機能としてリリースされており、各機能を連携させて複雑な動作を実現させる。

※ ただし、リリース当初は各機能の制御機能が実装されていない。

未知への対応

未知への対応

全ての機能は「どんな状況にでも対応できるよう」に

プリセットされた動作を持たない。

ただし、対応可能であるように拡張性がある。

ファーストリリース(新生児)

実装済機能
肺呼吸（リリース時に臍帯からの酸素提供から切替。事前検証無し）
アラート機能(log機能なし)
哺乳機能(リリース時に臍帯からの栄養補給から切替。事前検証無し)
消化器官(リリース前に検証済)
スリープ機能(リリース前に検証済)
排泄機能(リリース前に検証済)
原始反射(外からの刺激に反射的に体が動く機能。吸啜反射、口唇探索反射、引き起こし反射、把握反射、歩行反射、モロー反射など)
ゼネラルムーブメント(無意識に一定リズムで体を動かす機能)

乳児の成長

生後1ヶ月	生後3ヶ月	生後4ヶ月	生後5ヶ月	生後6ヶ月
新生児期間終了		首が座ってくる。	寝返り機能(β) リリース	寝返り機能リリース
体重が出生時より１～２kg増加し皮下脂肪が増え、 よくイメージされる「赤ちゃん」になる。	体重が出生時のおよそ倍まで成長する。	大脳が発達し「満腹機能(β)」がリリースされる。	聴覚と視覚の関連付けが始まり、 音のした方に顔を向ける様になる。	大脳が成長し好奇心、や記憶力が著しく成長する。
昼夜の判別機能 未実装。	昼夜の判別機能(β)が実装される。			「いないないばぁ」や「たかいたかい」を喜ぶ様になる。
哺乳時にゲップをさせないと戻してしまう不具合は解消されていない。	哺乳機能完成			

機能実装マトリクス

	嬰児	新生児	3ヶ月	半年	1年
頸骨の安定	×	×	○	○	○
腰骨の安定	×	×	○	○	○
寝返り	×	×	×	○	○
座る	×	×	×	○	○
ズリバイ	×	×	×	○	○
ハイハイ	×	×	×	○	○
掴まり立ち	×	×	×	○	○
つたい歩き	×	×	×	×	○
静歩行	×	×	×	×	○
動歩行	×	×	×	×	×

機能実装マトリクス

	嬰児	新生児	3ヶ月	半年	1年
頸骨の安定	×	×	○	○	○
腰骨の安定	×	×	○	○	○
寝返り	×	×	×	○	○
座る	×	×	×	○	○
ズリバイ	×	×	×	○	○
ハイハイ	×	×	×	○	○
掴まり立ち	×	×	×	○	○
つたい歩き	×	×	×	×	○
静歩行	×	×	×	×	○
動歩行	×	×	×	×	×

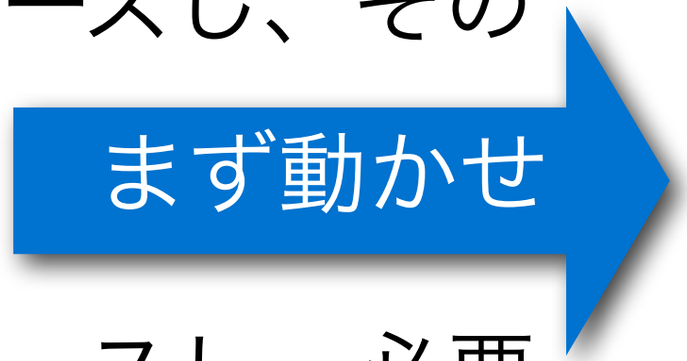
逐次機能リリース

ポイント

- 全ての機能は**開発完了**と同時にリリースし、その後も機能追加と改修を続ける。
- 開発の終わった単機能から順次リリースし、必要に応じてコントローラ機能をリリースする。
- 練習・訓練により大人が驚くような行為を軽々とやってのけるようになる。

ポイント

- 全ての機能は**開発完了**と同時にリリースし、その後も機能追加と改修を続ける。



- 開発の終わった単機能から順次リリースし、必要に応じてコントローラ機能をリリースする。

- 練習・訓練により大人が驚くような行為を軽々とやってのけるようになる。

ポイント

- 全ての機能は**開発完了**と同時にリリースし、その後も機能追加と改修を続ける。

まず動かせ

- 開発の終わった単機能から順次リリースし、必要に応じてコントローラ機能をリリースする。

簡潔に・単純に

- 練習・訓練により大人が驚くような行為を軽々とやってのけるようになる。

ポイント

- 全ての機能は**開発完了**と同時にリリースし、その後も機能追加と改修を続ける。

まず動かせ

- 開発の終わった単機能から順次リリースし、必要に応じてコントローラ機能をリリースする。

独立性と接続性

簡潔に・単純に

- 練習・訓練により大人が驚くような行為を軽々とやってのけるようになる。

ポイント

- 全ての機能は**開発完了**と同時にリリースし、その後も機能追加と改修を続ける。

まず動かせ

- 開発の終わった単機能から順次リリースし、必要に応じてコントローラ機能をリリースする。

独立性と接続性

簡潔に・単純に

- 練習・訓練により大人が驚くような行為を軽々とやってのけるようになる。

未知への対応

ポイント

- ユーザーの環境に応じた機能実装が可能
- 各機能を平行して開発・実装が可能
- 本能というコントローラーが最低限の連携を実現する。

環境:地域別実装

東北・北海道	寒冷地仕様（寒さに対する耐性の他、冬季の保存食として塩分濃度の高い食品を多用するために塩分に対する耐性を持つ）
秋田	日照時間が少ないので効率的にビタミンDを生成するために肌を白くする
東京	他地方に比べて空気が汚いので少ない酸素を有効に取り入れられる肺機能を持つ
沖縄	熱暴走に耐性。ただし湿度の伴う暑さに弱い
•	•
•	•

環境：言語

- 世界のいかなる言語にも対応できる様に、プリセットの言語を持たない。

- 周囲の利用する言語を半年～1年で理解、リリースから2年後には利用開始。

- 習得に個体差はあるものの概ね3年で周囲との音声による意思疎通が可能に。

環境：身体能力

- 坂道（山道）が多い地域では蹴り足を強く。
- 平坦な土地では膝を曲げない。
  - ※歩行時に足の裏が見え{る|ない}
- マニュピレータのプリセットが無いので、あらゆる食器（箸・フォーク・スプーン・ナイフ）に対応可能。

中締め

- Unix的思想は禅問答の様に普遍的な経験則の塊である。
- コンピュータシステムだけに限らず、あらゆる事象をより良くするための助けになる。
- 何の脈絡もない育児とUnixという2つの事柄を例示してその「普遍性」が伝わったなら…

引き続き

Unixの哲学を学ぶ？

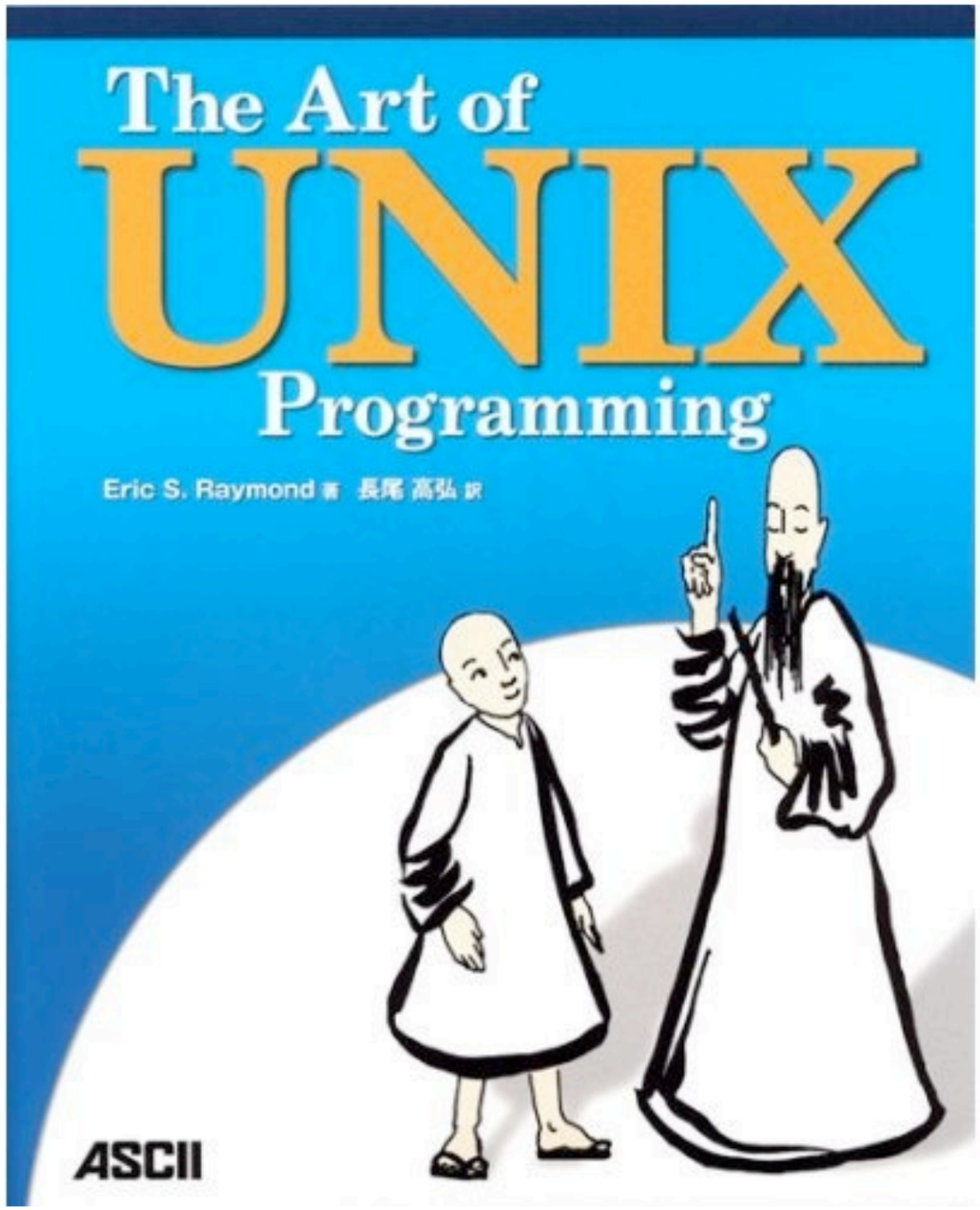
＞お終い

続きを見る

導師からの口伝

Unix文化に口伝でのみ継承されてきた「原則」をまとめた本があります。

The Art of UNIX Programming



- **Eric S.Raymond** (著), 長尾 高弘 (翻訳)
- 大型本: 560ページ
- 出版社: アスキー (2007/6/19)
- 言語 日本語
- **ISBN-10:** 4756149480
- **ISBN-13:** 978-4756149480
- 発売日： 2007/6/19
- 商品の寸法: 23.4 x 18.6 x 3.2 cm
- ￥ **6,090**

ちなみに

英語版は**無料**で公開されてます。

http://www.faqs.org/docs/artu/

Unix思想の基礎

Doug McIlry(Unixパイプの発明者)の言葉

「1つの事をきちんとするプログラムを書け」

「他のプログラムと協力できるプログラムを書け」

「普遍的なインターフェイスであるテキストストリームを処理するプログラムを書け」

Unix思想の基礎

(I) 個々のプログラムは1つのことをしっかりやるようにせよ。

新しい仕事をするなら、新しい機能を追加して古いプログラムを複雑にするのではなく、まったく新しいプログラムを作れ。

Unix思想の基礎

(II) すべてのプログラムの出力は、他の未知のプログラムの入力になるのだということを予想せよ。

余分な情報で出力をごちゃごちゃにしてはならない。縦方向の並びを厳密に揃えたり、バイナリ入力形式を使ったりすることを避けよ。対話的な入力にこだわるな。

Unix思想の基礎

(III) オペレーティングシステムであっても、ソフトウェアは早く（出来れば数週間以内に）試せるように設計、構築せよ。まずい部分は捨てて作り直すことを躊躇するな。

Unix思想の基礎

(IV) 技能の低い者に手伝ってもらうくらいなら、プログラミングの仕事を明確にするためにツールを使え。

ツールを作るために遠回りしなければならない場合でも、使ったあとで一部のツールを捨てることになることがわかっている場合でも、ツールを使え。

Unix思想の基礎

1.モジュール化の原則：

クリーンなインターフェイスで結合される

単純な部品を作れ。

適切なインターフェイスで結び付けられた単純な部品からシステムを作り上げれば、ほとんどの問題は局所化し、全体を壊さずに部品だけを改良することも可能になる。

Unix思想の基礎

2.明確性の原則：

巧妙になるより明確であれ。

パフォーマンスを少し上げるために、コードの複雑さを引き上げてはならない。

単に複雑なコードがバグを含みやすからだけではなく、将来のメンテナンスプログラマにとって読み取りにくくなる原因となる。

将来のメンテナンスプログラマが数年後の自分かもしれない事を考えて明確で穏当なコードを書くべきだ。

Unix思想の基礎

3.組み立て部品の原則：

他のプログラムと組み合わせられるように

作れ。

組み合わせられる事によって独自に車輪の再発明をせずにすむ。

他のプログラムの出力を自分の入力とし、自分の出力を他のプログラムの入力となるようにすれば、プログラムは自分の仕事に専念できる様になる。

1頁単位で出力を表示させるページャ機能を実装するよりも、パイプで繋いでmore(1)に渡せる様にする方が簡単で単純だ。

Unix思想の基礎

4.分離の原則：

メカニズムからポリシーを切り離せ。

エンジンからインターフェイスを切り離せ。

フロントエンドとバックエンドを切り離す事によって、2つのプログラム全体の複雑さは同じ機能を実装する1つのプログラムより かなりマシになる場合が多い。

Unix思想の基礎

5.単純性の原則：

単純になるように設計せよ。

複雑な部分を追加するのは、

どうしても必要なときだけに制限せよ。

流行に支配されて誰も使わない機能を盛り込んではならない。

自己の技術を誇示する為に「美しく入り組んだ構造物」を創り上げてはならない。

それは矛盾した存在であり、Unixの世界では「単純で美しい」事を美徳とする。

Unix思想の基礎

6.倹約の原則：

他のものでは代えられないことが

明確に実証されない限り、

大きなプログラムを書くな。

「大きい」とはコードの分量と内部の複雑さの両方の意味。

プログラムの分割に失敗した時だけ、大きなプログラムを書くのである。

Unix思想の基礎

7.透明性の原則：

デバッグや調査が簡単になるように、

わかりやすさを目指して設計せよ。

少なくとも「デバッグオプション」は付け足し程度の後知恵であってはならない。

透明性と開示性を目標に内部状態を監視、表示できるプログラムを書くのだ。

Unix思想の基礎

8.安定性の原則：

安定性は、透明性と単純性から生まれる。

構造が単純で、何が起きているのかすぐに分かるプログラムは、人間の頭でもすべてをまとめて理解できる。

プログラムの内部構造について正しく説明できるのであれば、そのプログラムが正しいかどうかの保証も容易く、壊れていてもすぐに直すことが出来る。

Unix思想の基礎

9.表現性の原則：

知識をデータのなかに固め、

プログラムロジックが

楽で安定したものになるようにせよ。

データはプログラムロジックよりも御し易い。

だから、データ構造を複雑にするか、コードを複雑にするかを選ばなくてはならなくなったら、迷わず前者を選ぶべきだ。[40]

Unix思想の基礎

10.驚き最小の原則：

インターフェイスは、

驚きが最小になるように設計せよ。

もっとも簡単に使えるプログラムは、ユーザーが新しく学ばなければならないことが
　　もっとも少ないプログラムのことだ。

一見似ているが微妙に異なる動作をするインターフェイスは避けなければならない。

誤った期待を高める事は、裏切られた時の失望も大きくなる。

Unix思想の基礎

11.沈黙の原則：

どうしても言わなければならない想定外な事がないのなら、プログラムは何もいうな。

「沈黙は金なり」

よく考えて設計されたプログラムなら、ユーザーの注意力や集中力は、どうしても必要なとき以外は使うべきでない貴重で限りあるリソースとして取り扱う。

Unix思想の基礎

12.修復の原則：

エラーを起こさなければならないときには、

できる限り早い段階で

けたたましくエラーを起こせ。

通常の処理だけでなく、失敗するときも透明でなければならない。

「受け入れるものは寛容に、送るものについては保守的に」

ただし、寛容であるべきものは仕様であり、43仕様の解釈ではない。

Unix思想の基礎

13.経済性の原則：

プログラマの時間は高価だ。

マシンの時間よりもプログラマの時間を節約せよ。

プログラマの時間を節約する為に高水準言語や統合開発環境を使うだけでなく、
プログラミングの低水準の仕事を多くこなすためにプログラムを書くのだ。

Unix思想の基礎

14.生成の原則：

手作業のハックを避けよ。

可能なら、プログラムを書くための
プログラムを書け。

人間は細かい仕事が苦手だ。

だからこそ、コンパイラやインタープリタが作られたのだ。

Unix思想の基礎

15.最適化の原則：

磨く前にプロトタイプを作れ。

最適化する前にプロトタイプが動くようにせよ。

「今提供できる90%の機能は、永遠に提供できない100%の機能よりもはるかに良い」

「まず動かし、正しくして、速くせよ」

Unix思想の基礎

16.多様性の原則：

「唯一の正しい方法」とするすべての主張を

信用するな。

設計者の想像力を超えてソフトウェアを設計することは出来ない。

すべての用途を予測して設計できないのであれば、オープンで拡張性の高い、

あちこちにカスタマイズのフックをつけたシステムを設計するべきだ。

Unix思想の基礎

17.拡張性の原則：

未来は予想外に早くやってくる。

未来を見すえて設計せよ。

将来の開発者たちがコードを全部捨ててアーキテクチャを作りなおしたりせずに、

新しい機能を追加できるように作るべきだ。

ジョイントを柔軟なものにして、後で必要になった時に機能を簡単に柄できるよう

に書いておこうというアドバイスだ。

Unix思想まとめ

- K.I.S.S.
- Keep It Simple,Stupid!
- 「単純を保て、愚か者よ」