漸進的情報処理

漆畑 晶

漸進的情報処理

UNIX が生まれたベル研究所

ベル研究所(Bell Telephone Laboratories)はアメリカ電信電話会社(AT&T)の研究子会社として 1925 年に誕生しました。1921 年に AT&T はアメリカ合衆国での電信電話事業の独占的運営権を入手しましたが、それと引き換えに料金は政府が決定する公定価格となりました。価格を上げられなくても利益を増やす方法はあります。それは各種部品の品質と信頼性を高めてコストを削減することでした。広大な国土に電信網を展開していたので部品寿命や故障率の改善は利益に大きく寄与しました。さらに電話の信頼性と音声品質が高まると人々は電話を頻繁にかけるようになり、売上は増加、やはり利益が増えます。

ベル研究所は優秀な数学者、科学者、技術者を雇い、基礎研究、実用研究、装置機器を開発改良ともに行いました。必要に応じて専門的な測定機器を開発し実験を行いました。AT&T は信用があり、経営も安定していたので優秀な人材を集めることができました。ノーベル賞受賞者を含む各分野の代表が集う、アメリカ屈指の研究所でした。

ベル研究所では品質を測定するための指標や手法が確立されました。音量の単位のデシベル(dB)はベル研究所で生まれました。今では音量、特に騒音の単位として知られていますが、元は電話線の両端の間でどれだけ音声信号が変化（減衰）したかを測るために考え出されたものでした。今日各産業で使われている統計的品質管理と管理サイクルはベル研究所所属のウオルター・シューハート(Walter Shewhart) 博士が考案し、その弟子のエドワーズ・デミング (Edwards Deming)博士が発展普及させたものです。デミング博士の教えを特に熱心に聞いたのが日本の産業界でした。戦後日本の工業製品の品質が向上し、各産業で生産性が伸びましたが、ベル研究所で生まれた品質管理手法を導入していなかったら歴史は大きく違っていたでしょう。

第二次世界大戦後の 1947 年、ベル研究所では半導体トランジスタが発明されました。増幅回路には真空管が不可欠でしたが、これは故障が多く、故障しにくい部品に置き換えたいとの要請に応えたものです。トランジスタによって品質の高いデジタル通信が可能になり、また交換機は機械式から無接点の電子式に置き換わります。デジタル通信と一種のコンピュータとも言える電子式交換装置の開発のためにベル研究所は情報工学の専門家も登用しました。この専門家達が研究用に開発したのが UNIX です。

UNIX が誕生したのは 1969 年とされます。この頃はコンピュータの性能が向上して、プログラミング言語が進化していました。高級言語が発達し機能が強化されて行きます。また専門家でなくても使いやすい言語として BASIC が登場して学生の間に普及します。実用的な応用ソフトが登場しましが、機能が充実すると必然的に大きくなって行きました。ベル研究所の専門家達はこの傾向に乗らず、むしろ正反対の方向を探究しました。プログラムはなるべく小さくし、それを柔軟に結合して力を引き出そうと考えました。この情報処理に対する基本的な姿勢は「UNIX 哲学」と呼ばれます。その中心にあったのはダグラス・マッキルロイ(Douglas McIlroy)博士です。 ***Do one thing and do it well.*** 「単一の機能に徹し、それをよくこなすべきだ」と言ったのはこのマッキルロイ博士の言葉です。彼は echo をはじめいくつかのプログラムを書いていますが、最も重要な功績は入出力を結合するパイプの発明です。

現在 UNIX 系統の OS が世界で広く使われています。インターネットとそれに接続される商業、金融、報道、学術は UNIX 無しでは考えられません。電話会社の研究機関の開発だったので UNIX は最初から通信に強い OS でした。またベル研究所の伝統的な品質へのこだわりが反映され、UNIX には品質管理のためのいくつかの工夫が施され、その使い方、考え方が利用者達によって継承されて行ったのが普及した理由だと思います。

UNIX の特徴本質への関心の低下

1980 年代に UNIX は普及して、多くの大学や企業で使われるようになりました。一方でその特徴的な設計思想を意識しない利用者が増えて行きました。それまで FORTRAN 等他の言語を主に教えていた大学の教員には UNIX の特質を把握しないまま教える人がいました。UNIX の教書にも同様にコマンドを羅列するだけのものが出版されました。世界的に普及してもなお、UNIX の特徴的な思想が一部にしか理解されていない状況は続いていると思います。(＊１）

1980 年代には AT&T とベル研究所以外の業者も UNIX ワークステーションの製造販売に乗り出し、使いやすさの向上を目指してグラフィック・ユーザー・インターフェイス（GUI）を充実させました。GUI が普及するとアプリケーションの起動はアイコンのクリックで行い、コマンドラインを常用しない人が増えました。

また 1990 年代には Linux が普及しますが、作成者リーナス・トルバルズ氏はカーネルを作っただけなのに「UNIX 互換の OS を作成した」と主張しました。（＊２）カーネルは OS の最重要部分には違いありませんが、カーネルだけでは UNIX の特徴的な操作（オペレーション）はできません。人の操作を受け入れるのはシェルです。（＊３）トルバルズ氏の主張が広く支持されたというのはその時期までに UNIX の本質はどこにあるのか曖昧になっていたのが一因と言えます。（＊４）

漸進的情報処理の位置づけ

UNIX は要素分解、ソフトウエア資源の再利用、可搬性などが大きな特徴です。AT&T では電話網全体を一つの巨大な機械（システム）と見なしていました。全体の品質を良くするには個々の部品の品質を高め、信頼できるパーツを積み上げて行くとする思想がありました。この考えは c 言語は構造化による要素分解によく反映されています。構造化はその後のプログラミング言語に大きな影響を与えました。

漸進的情報処理も UNIX の特徴です。一種の要素分解です。この用語は聞き慣れないものと思います。漸進とは「徐々に進む」という意味です。英語では Incremental programming と言います。

漸進的情報処理の例

<u>例１：ファイルの一括削除</u>

漸進的情報処理とは何か理解してもらうために極めて簡単な例を考えてみましょう。

コンピュータの中に散在する、ある条件に合致したファイルを一括して削除したいとします。こういうのは慌てて実行すると痛い目に遭うものです。削除すべきでないファイルを誤って消してしまったという経験、誰でもあると思います。そうは言っても、一つ一つ消去するのは面倒です。面倒な操作は疲れるので、これはこれで間違いを犯すものです。

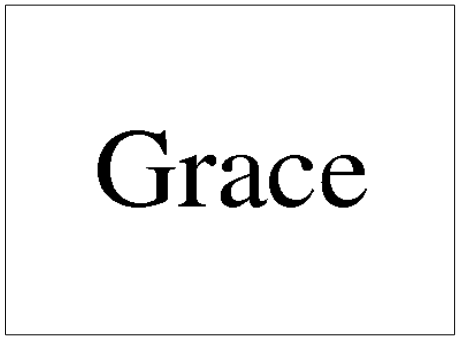
この問題を解決する方法は削除すべきファイルをまず列挙だけして、よく確認してから改めてファイルを削除する動作を行うことです。

漸進的情報処理とは課題を小分けにして、確認しながら、時には調節を入れながら作業を進めるものです。

例２：陰をつける画像処理

Netpbm は UNIX の基本理念に忠実な画像処理の道具の集合体です。一つのプログラムは一つだけの用途をこなします。プログラムの結合にはパイプを使います。

まず英数字その他の文字をレンダリングして画像を作成します



ぼかします　(左）
ぼかした画像の上と左に白い余白を加えて平行移動します　（右）

元の字の色を変えます。ここでは淡い灰色とします　（左）
灰色の文字の画像とぼかしを入れた画像を重ね合わせると字に陰がつきます　（右）

一連の処理は単一機能のプログラムを逐次実行することによって実現しています。この処理全体はよく利用されるので ppmshadow という名のプログラムなっています。

実際に Netpbm は以下のようなパターンで利用されます。単一画像の編集には GIMP など使いやすい対話型のソフトウエアがありますから、複数画像を一括処理したい場面で使われます。

受信 →デコード → 編集加工 → エンコード→送信
受信 →デコード → 編集加工 → 分析・画像認識
受信 →デコード → 編集加工 → 比較
新規作成（文字レンダリング） → 編集加工 → エンコード→送信

編集加工の基本的な操作として拡大縮小、明暗の調節、回転などが提供されています。この段階は複数のプログラムの結合による場合もありすし、無い場合もあります。パラメータを調節しながら既成のプログラムを当てて、不十分な場合は利用者が自作します。Netpbm のフォーマットは単純なので自作は難しくないとされています。

Netpbm は基本的な道具を提供するのが目的です。例えば GIF から JPEG へ変換するプログラムはあったら便利でしょうが、**Netpbm にはありません。**（＊３）デコーダとエンコーダは分かれており、利用者が必要に応じて組み合わせて機能を実現します。

例３：エラーメッセージの分類

次はプログラムのテストを行ないエラー終了したケースです。エラーメッセージが多数表示されていますので、これを漸進的に処理します。

（1）テストの出力：

```
$ make -C check

== bmp-quant-roundtrip.test ==
pnmcolormap : command not found in test path
pnmcolormap failed, rc=22528
ppmtobmp : command not found in test path
bmptopnm : command not found in test path
bmptopnm : command not found in test path
ppmtobmp : command not found in test path
pnmcolormap : command not found in test path
pnmcolormap failed, rc=22528
ppmtobmp : command not found in test path
bmptopnm : command not found in test path
bmptopnm : command not found in test path
ppmtobmp : command not found in test path
bmp-quant-roundtrip.test: FAILURE

== gif-quant-roundtrip.test ==
pnmcolormap : command not found in test path
pnmcolormap failed, rc=22528
gif-quant-roundtrip.test: FAILURE

Test summary:
==================
FAILURE 2
TOTAL TESTABLE 2
==================
```

（2）文字列 command のある行だけを抽出：

```
$ make -C check > test.out; grep "command" test.out

pnmcolormap : command not found in test path
ppmtobmp : command not found in test path
bmptopnm : command not found in test path
bmptopnm : command not found in test path
ppmtobmp : command not found in test path
pnmcolormap : command not found in test path
ppmtobmp : command not found in test path
bmptopnm : command not found in test path
bmptopnm : command not found in test path
ppmtobmp : command not found in test path
pnmcolormap : command not found in test path
```

（3）さらに各行最初の語だけを抽出：

```
grep "command" test.out | awk '{print $1}'

pnmcolormap
ppmtobmp
bmptopnm
bmptopnm
ppmtobmp
pnmcolormap
ppmtobmp
bmptopnm
bmptopnm
ppmtobmp
pnmcolormap
```

（4）辞書順にソート：

```
grep "command" test.out | awk '{print $1}' | sort

bmptopnm
bmptopnm
bmptopnm
pnmcolormap
pnmcolormap
pnmcolormap
ppmtobmp
ppmtobmp
ppmtobmp
ppmtobmp
```

（5）重複をまとめる：

```
grep "command" test.out | awk '{print $1}' | sort | uniq

bmptopnm
pnmcolormap
ppmtobmp
```

一応これで終わりにしますが、さらに続けることもできます。テストはこの３つのコマンドが実行できなかったために失敗と判明しました。これは実行形式ファイルの欠如か、PATH の問題か。原因究明と対処のためにさらにこの末尾に命令を逐次追加して行くこともできます。

さてここまで命令を並べてみましょう

```
 grep "command" test.out
 grep "command" test.out | awk '{print $1}'
 grep "command" test.out | awk '{print $1}'
 grep "command" test.out | awk '{print $1}' | sort
 grep "command" test.out | awk '{print $1}' | sort | uniq
```

UNIX ではこのようにして、全体の命令がだんだん長くなっていくのが通例です。単一のコマンドではなく、いくつかを組み合わせて発することが日常的に行われます。このような命令の並びを**パイプライン**と呼びます。パイプラインは漸次伸長して望みの処理が実現します。シェルはアプリケーションを起動するにとどまらず、**対話的にプログラミング**するための道具でもあります。

このように欠損ファイルのリストを生成するのならどんなプログラミング言語で可能です。しかしエラーメッセージのようにどんな入力が来るか予想できない場合、事前にプログラムを用意するわけには行きません。データを見てから、少しずつ絞り込み、様子を見ながら調整を加えていく、そんな作業の進め方が有効です。

パイプラインは単線構造なので理解しやすいのが長所です。他人から譲受してもらって、一見して機能がわからない場合は最初（左）から一段階ずつ順次に実行して効果を目で確かめます。

構造化以前の言語 FORTRAN BASIC COBOL では goto 多用による、いわゆるスパゲティ・コードが見られました。 c 言語が定着させた構造化によってこの問題は克服されました。それでも c 言語等で関数を使うとソースの記述順と実行順は異なったものとなります。単線構造は実行順序が明確です。単線構造で解ける課題は意外と多いのですが、探求しないとこれはわかりません。上記例 3 のエラーメッセージの整理のような例題をたくさん解いて感覚をつかむ必要があります。漸進的情報処理は汎用性のあるスキルで多くの局面で活用が可能、心がけ次第で経験を積むことができます。

漸進的情報処理と安全な操作

安全な操作のコツがあるとすればそれはまず第一に「絶対安全なるものは存在しない、人の操作は間違いがあり危険を伴うと念じておく」ことでしょうが、その次にするべきは「確認しながら作業を進める」ことでしょう。構造が複雑だと確認は困難になります。できるだけ単純な構造が望まれます。

コマンドにはデータを上書きしたり消去したりするコマンドとそうでないコマンドがあります。その違いを意識すべきです。UNIX ではプログラムは一つの機能しか果たしません。何をするか明確、何をしないかというのも明確です。老荘思想でみるような逆説ですが、**余計なことをする道具は使いにくく、万能の道具は余計なことばかりするのです。**

人は強く便利な道具を求めるものですが、安全な操作のためには**弱い道具を使う**のがコツです。

漸進的情報処理への批判

漸進的情報処理では複数の単純なコマンドを結合します。複数の道具の使い方を覚える必要がありますし、単純な道具ほど習熟が難しいものです。シェルは端末プログラムであるとともに、プログラミング言語でもあります。一見簡単な処理でも、計算や場合分けを要する場面もあります。こうした状況で使われるのは上記の例で登場する awk です。この awk は習得は容易ですが一人前の言語です。その他、構築作業を管理する make も機能が多く Makefile はプログラムです。UNIX を上手に活用するには、c、シェル、awk、make の他に多数のコマンドの使い方を習得する必要があります。

「UNIX の多言語は大きな問題点」と考えた人達は万能スクリプト言語を開発と使用を推奨しました。最初に人気が出たのは Perl です。Perl をはじめスクリプト言語 c の構造化の考え方を踏襲しています。複数の Perl スクリプトをパイプラインで結合して漸進的処理を行うことは可能ですが、実際にそのような使い方はほとんどされず、一つのスクリプトで一連の処理を全部担当するのが多いと思います。またパイプラインでは出力はふつう標準出力（端末表示）ですが、スクリプトでは入力と出力が同じファイルというのがよく見られます。入力を上書きするのは便利な機能ですが危険を伴います。

複数のコンピュータ言語をマスターできるのは優秀な人です。多数の言語の習熟を期待する点、UNIX にはエリート主義が見られます。それだけではありません。マッキルロイ博士は、未熟な見習い技術者に仕事を任せるより、優秀な人が道具を有効活用してコンピュータに仕事をさせた方が良いと言っています。現実の企業では多数の未熟練プログラマが働いていますが、マッキルロイ博士の方針に従うとこの人達にさせる仕事がなくなってしまいます。情報処理技術者は現場に出て仕事をしながら技術を習得する（職場内訓練、On the Job Training、OJT）より、一定の技倆を身につけるまでは仕事につかず（職場外訓練、Off-JT）学習に専念すべきだということになりますが、求められる「一定の

技術水準」は高いと言えます。

現在主流となっている対話型のエディタ、ワードプロセッサ、表計算、グラフィック・エディタは操作がすぐ画面上に反映され、ある意味、漸進的情報処理を更に進めたとも言えます。しかし文字列の置換等、一括処理機能もありますが、柔軟性はパイプラインに及びません。コンピュータの力を活用しない手作業が多いのが現実です。しかし初心者にもわかりやすく、一般の技術者に使いやすいもので広く使われています。

対話型のエディタや開発支援ソフトの使用に習熟した人なら漸進的情報処理はすぐ飲み込めるかと言うと、そうでは無いようです。上記のように、万能の便利な道具を排し、なるべく弱い道具を利用するというのは価値観の改革を必要とします。普段の作業方法とは全く違った方針を唱える人から学ぶには礼が欠かせません。

品質管理サイクル、科学的方法との関係

統計的品質管理を提唱したベル研究所のシューハート博士は製造業における管理サイクルを提唱しました。弟子のデミング博士はこれを発展させ普及させました。デミング博士は Plan (計画) Do (実行) Study (検証) Act(改善) の順に行い、これを循環させて経験を蓄積し、管理を充実させていくと唱えています。

計画、実行、検証、改善の順で何かを進めると言っても、課題が大きくなるとその通りに実施するのは困難になります。大きな課題を小さく分けていくのが効果的です。UNIX に見られる漸進的情報処理とはまさに大きな課題を小さな物に分けるものです。他の処理系でこういうことが全くできないかというと、そうでも無いでしょうが、UNIX のシェルと周辺ツールは漸進的処理を設計段階から意識して作成されているのが強みだと言えます。

生産現場で計画、実行、考証、改善の循環によって品質管理を行う考えはガリレオの提唱した仮説、実験、検証のプロセスを工学に応用したものです。ベル研究所では情報通信の専門家ばかりでなく、物理、化学、数学の専門家も活躍していて専門領域を越えた意見交換が奨励されていました。また、ベル研究所では、小さな実験でもきっちりやり遂げる人達を尊重する伝統がありました。

科学実験を成功させるには条件を限定して他の要因が入り込まないようにするのが肝要です。この方針を生産現場に応用すると小さな部品までも検査をしっかり行うことになります。情報処理に応用すると関数の単体テストの徹底やそのつど確認しながら進める漸進的情報処理になります。

情報技術者は自分の仕事が特殊だと考える傾向があります。木工、陶芸、鍛冶など伝統工芸には作業のミスを防止し、製品品質を向上させるための伝承の技があるものです。情報産業はこの点においてまだ未熟で、他の分野を参考にするのも不得意です。技術の発展史、他産業の熟練技術者の育成をも視野に入れ、合理的な品質向上策と生産性向上を考えるべきと考えます。

2019 年 2 月　漆畑晶

注

（1）最近では UNIX 系 OS 用の応用ソフトにも UNIX の思想からかけ離れた物がある。動画編集の ffmpeg が一例。

（2）「Linux はカーネルだが、それだけで OS」との説の他、「カーネルとライブラリ、シェル、基幹コマンド等他のソフトウエアを合わせて Linux OS」とするという見解がある。Wikipedia の記事は後者の立場。

（3）「カーネル」kernel と「シェル」shell は対語で、麦や米などの穀類の種子の粒（米なら玄米）と外殻（籾殻）を言う言葉が起源。カーネルはプロセス管理、ハードウエア資源の割当を行い、シェルは人の入力に対応する。

（4）UNIX という語自体、専門家の間で様々に解釈されている。「昔流行った OS」「BSD UNIX として細い命脈を保っている」との見解が聞かれる。一方でベル研究所の UNIX の開発者達は Linux は UNIX の伝統を継いでいると見ているよう。

（5）もしカーネルだけで OS とするなら、Netpbm はアプリケーションになる。ところが Netpbm はプログラムを作るための部品集なので一般の利用者が直接使うための配慮がなされていない。一般利用者にとっては使いずらい物をアプリケーションに分類するのは無理がある。

参考文献

ベル研究所について

世界の技術を支配するベル研究所の興亡
ジョン・ガートナー著　土方奈美訳

ベル研究所の研究と親会社 AT&T の事業方針、合衆国の政策の関係を詳しく解説。人物中心の本。クロード・シャノン博士の人となりと業績、半導体トランジスタの発明、ベル研究所を去った研究者がカリフォルニアの現在シリコンバレーと呼ばれる地で半導体製造業を起業した様子など詳しい。トランジスタから通信衛星、携帯電話まで、今は身の回りに普通に見られるようになった技術のいかに多くがベル研究所由来かと知って驚く。結果的に失敗に終わった研究についても触れていて興味深い。

統計的品質管理について

デミング博士の新経営システム論
産業・行政・教育のために
W・エドワーズ・デミング著　NTT データ通信品質管理研究会訳

統計的品質管理を戦後日本に伝えたのはデミング博士だが、彼が教えたのは統計学の技法にとどまらない。企業と労働者、企業と企業のあるべき関係についての主張はいわゆる日本的経営に取り込まれ、経済成長の基礎となった。本書は日本で成功した手法をアメリカの企業経営者、管理職に伝えようと記されたもの。誰でも簡単にできるおはじきの散布実験を通して「調整し過ぎると、却って品質が悪化する状況」を紹介し、特殊要因と一般要因を区別する事の重要性を上手に解説している。

UNIX の特徴的操作について

ソフトウェアの道具箱
A. ロビンズ著
https://linuxjm.osdn.jp/info/GNU_coreutils/coreutils-ja_210.html#Opening-the-software-toolbox

短い文ながら、UNIX の思想に沿った小さなプログラムの組み合わせ処理を端的に解説している。

著者紹介

漆畑 晶

プロフィール

- 画像処理ソフトウェアユーティリティ群 Netpbm の主要開発者
- FSF 創設者のリチャード・ストールマン博士と親交あり
- 中国の大学（上海復旦大学、南京大学、天津大学、中国科学院大学等）で講演多数

信州大学経済学部在学中に「国富論」やアメリカ産業史の本に加えて、「論語」「老子」などの古典を読んで引用する力を磨く一方で、R 言語による多変量回帰分析プログラムや、駅すぱーとの核心部である最短距離経路探索のプログラムを自作していました。

オープンソース・ソフトウェアについて

なにか良いものをタダで貰ったり使わせてもらったときは、必ずお礼を言うものです。お礼を欠いていると、おかしなことになってしまいます。誰でも自由に利用できるオープンソース・ソフトウェアも同じです。感謝を欠かしてはならないのです。

漸進的情報処理

著　者：漆畑 晶

発行者：（特非）オープンソースソフトウェア協会
http://www.ossaj.org/
info@ossaj.org

発行日：2024年2月1日