

UI とキーボード&マウスとウィンドウシステム

2023/02/21, 04/25 山田 泰司 <taiji@aihara.co.jp>

筆者は GUI と CUI の間に最適なソリューションがあると信じ、機会ある毎に様々なコーディングを試して来た。それはさて置き、GUI と CUI の歴史を紐解き学ぶ機会が得られ、記すべき知見に出会ったので、ここに初学者にも伝わるように纏める。

目次

- GUI の非効率さについて
- CUI の活用について
- ポインティングデバイスの非効率さについて
- ショートカットキーについて
- ウィンドウのフォーカスと配置について
- CUI と可視化のためのウィンドウ
- Unix 文化の継承者
- Borland Delphi の継承者
- メインフレーム・ミニコン文化の継承者
- ポチクリ文化の継承者たち
- /home/\$LOGNAME — NFS 文化の継承者

GUI の非効率さについて

一つ目の事実を、まず紹介する。

GUI の非効率さを先見的に見抜いていたのは Apple Macintosh 開発者の一人、ジェフ・ラスキンである。

氏は所謂ショートカットキーを、Macintosh のグラフィカルなメニューに導入した。そして、スティーブ・ジョブスらは、それを排除しなかった。

マイクロソフトは Macintosh 向けのアプリケーション開発を通じて GUI を自ら実装することになる。ショートカットキーは彼らの Windows にも伝承された。加えて、不格好な、アレンジされたショートカットキーも導入され、幸いにそれは mac には逆輸入はされていない、ファイルメニューの「New」、日本語だと「新規(N)」とかいう国際化に配慮しないアレである。正確にはこれはショートカットキーとは言えないが、キーボード操作のみでメニューからのアイテムに 1 文字で辿れる工夫である。重ね々、Macintosh 及び macOS には、この不格好な工夫は伝承されずに幸いである（これがあるので Windows ではショートカットキーが整備されない原因である）。

CUI の活用について

さて、CUI を効率的に駆動する文化はやはり Unix と、それをオープンソースとして維持して来たのは GNU プロジェクトに他ならない。そして、BSD Unix や Solaris におけるビル・ジョイ¹が `csh`、`tcsh` そして `bash` や `zsh` へと、ターミナルのインタラクティブ環境に与えた貢献が大きい。

`bash` や `zsh` のような `ksh` (Korn Shell) つまり POSIX シェルが重要である理由は、作業の再現性と伝達のし易さの確保にある。二つ目の事実は以下である。

とは言っても、`csh` 系の `tcsh` など使ってはならない（これは調べれば分かるので割愛）。

一方、マイクロソフトは貧弱な DOS プロンプト環境及びバッチファイルの問題を解決すべく Windows Scripting Host (WSH) を開発した。GUI だけではヒューマンインタフェースは非効率であることは少なくとも認識していた証左だが、WSH はインターネット・エクスプローラ (IE) と同じ過ちを犯し、そして、敬遠され、現在では `cmd.exe` と PowerShell が使えるレベルになっている。

貧弱とシンプルは異なるように、複雑と完全も異なる。WSH は「シンプル」では無かった。

KISS 原理:

“Keep It Simple, Stupid!”（単純なままにせえよ、ばか！） — 読み人知らず

この日本語訳や使われた場面の歴史的な詳細は分からないが、Unix 文化の一つという伝聞がある。筆者は、要求に応じて開発したプログラムを、最終的には削ぎ落とせるところはこそぎ落として、コードとコメントを最小限にする作業は必ず行う。過剰な機能があれば、退避しては置くものの、バッサリ捨てた上で伝達する。KISS とは、問題と解法の複雑さのバランスだろう。

その意味で、PowerShell は顧客に配慮し過ぎていて面倒なのである。使うが日常的に使えない。

ポインティングデバイスの非効率さについて

三つ目の事実を提起する。

GUI の非効率さの改善は mac よりも Windows の方が多少優れている。

GUI の非効率さはポインタの移動の為に、キーボードのホームポジションから手指が離れてしまうことにある。特にサーバ管理者ではマウスのようなポインティングデバイスはほぼ使わない。

¹ 1986 年 BSD Unix 開発の功績で ACM グレース・ホッパー賞受賞

しかし、mac にて既定の設定ではその「フルキーボードアクセス」がオフになっているのである。一方、Windows は無条件でポインティングデバイス無しでも（いさかステップを要するものの）完全にキーボードのみで操作が可能になっている。

mac でこれをオンにするには、環境設定 — キーボード — ショートカット — すべての操作をキーボードで行う、をオンにするか、**^F7** でトグルをオンにする必要がある。**それでもポインティングデバイスが必須になる場面が残されており（これはセキュリティポリシー的に致し方がないと思われる）、^F6 でフォーカスを移動できてもタブやカーソルで辿れないことも稀にある。**Windows ではそのようなことは無い。

実は Mac オペレーティングシステムのインストーラでは無条件にこれがオンになっている。ポインティングデバイス無しでもシステムがインストール出来るようにとの配慮なのだろう。

ショートカットキーについて

ところで、ジェフ・ラスキンは Macintosh の完成を待たずに Apple を退社し、数年後キヤノンから Canon Cat というパソコンを出来している。筆者は archive.org のエミュレータで体験してみたが、皆目使い方が分からない代物であった。スティーブ・ジョブスが居なかったらその後の Macintosh は Cat のようになっていたかも知れない。尤もその後の Mac は一時期は猫科 (Felidae) のコード名シリーズなのはラスキンに敬意を表してのものだろう。プログラミング言語 Swift の名も然りである（同氏の SwyftCard は BASIC による計算環境とワードプロセッサが融合した操作環境らしい、事実確認は私にはできない）。

そして、同氏によるスペースバーを挟む2つの Leap キーは **⌘** キーとして現在の mac にも脈々と受け継がれている。PC における Windows キーとは役割は根本的に異なり、むしろ PC におけるコントロールキーに相当するが、後者は**右側には存在しない**。Leap キーは Shift キーと同様に両側にあるべきだ。

ウィンドウのフォーカスと配置について

Windows と Mac の GUI デスクトップの欠点は、フォーカスされていないウィンドウを前面に配置できないことにある。かつての多くの X Window System (X11) のウィンドウマネージャではこれが出来るのが普通だったので、参照したいウィンドウを脇で眺めながら別のウィンドウの操作が可能であった。昨今は、タイル状にウィンドウを並べることでこの欠点を回避するらしい。

X11 経験者には需要があるらしく、Windows でも mac でも裏技的にこれを実現する方法があるので参考にされたい。

- ・ **^⌘ left click** … Mac
- ・ **レジストリを編集** … Windows

関連する扱いだが「ポインタをホバーしたウィンドウをフォーカス」する需要もある。これで前面にウィンドウを配置しないのなら、まさに古の X11 の挙動そのものとなる。

CUI と可視化のためのウィンドウ

四つ目の事実を提起したい。

ウィンドウは GUI のためだけのものではない。

かつて、`xdu` という X11 ソフトウェアがあった。これは `du` の階層ファイル容量の出力をグラフィカルに可視化するためのものである。コマンドラインからの利用が想定されていて、以下のようを使う。

```
% du -m . | xdu
```

かつて、`mgdiff` という Motif/X11 ソフトウェアがあった。これは2つのファイルの差分をグラフィカルに可視化するためのものである。コマンドラインからの利用が想定されていて、以下のようを使う。

```
% mgdiff filename.orig filename
```

他にも KDE や GNOME が台頭する以前には、こうしたコマンドラインからの利用を想定したウィンドウアプリケーションが多数存在していた。このようなアプリケーションにも大きな市民権を与えるべきである。

Unix 文化の継承者

五つ目の事実を述べる。

mac には Unix 文化が確かに継承されている。

`xdu` に相当する標準的なものは知らないが、`mgdiff` に相当するものは Apple 謹製 Xcode に同梱されている。以下のようを使う。

```
% opendiff filename.orig filename
```

mac を Unix 使いしてるなら分かるだろうが、Apple はコマンドラインをかなり大事にしている。NeXTSTEP 及び FreeBSD Unix 由来の OS なのだから、開発者らの文化で当然なのだ。

一方で、長らく CUI を蔑ろにしてきた Windows では PowerShell で巻き返しを図れたものの、そうした文化そのものが育たなかった。WSL (Windows Subsystem for Linux)² もしくは Cygwin を解決策と考えるなら時すでに遅しなのかも知れない。文化はユーザ自身がベンダーと共に育てていかなければならないのだ。

² かつての SFU (Services for UNIX) の代替。

「プロダクトに自分なりの文化を一つも抱いていない。そういうセンスのなさ」

— マイクロソフトについて、スティーブ・ジョブズ, PBS ドキュメンタリー, 1996.

Borland Delphi の継承者

Borland Turbo Pascal / Delphi の開発者であるアンダース・ヘルスバーグはマイクロソフトに移籍して C# / TypeScript / .NET Framework の開発に携わる。これらは Windows の共通言語基盤 (CLI: Common Language Infrastructure) の中核技術である。一方で、サン・マイクロシステムズの Java 環境はプログラミング言語が Java に縛られ、問題に最適な言語を選べず窮屈である。ビル・ゲイツ³所縁の BASIC 推しに辟易してしまう VBA となんら変わらない。その点 CLI は優れている。CLI は [ECMA-335](#) 及び [ISO/IEC 23271](#) として国際規格として標準化済みだ。

.NET は Mac や Unix でも [Mono プロジェクト](#)の成果で利用することができる。本家マイクロソフト謹製のオープンソースも提供されている。

六つ目の事実を述べる。

C# は優れた言語かつ国際標準 ([ECMA-334](#) 及び [ISO/IEC 23270](#)) だが、C++ の資産を一部でもそのまま使いたければ C++/CLI ([ECMA-372](#)) というありがたい選択肢もある。

メインフレーム・ミニコン文化の継承者

メインフレームとミニコン文化の定義はしづらいが広義には Unix 文化と重なる部分が多いだろう。狭義にすれば DEC 文化と言えるかも知れない。現在の Windows は元 DEC のデイヴィッド・カトラー⁴の設計なのでマイクロソフトは古の DEC 継承者と言っても過言では無いと思う。

しかし、ここで言いたいのは更に狭義のメインフレーム・ミニコン文化のことを指す。それは Excel に代表される「操作環境としてのスプレッドシート」である。

スプレッドシートはウィンドウシステムとは無縁な Apple II からの人気だったので Windows はもとより MS-DOS にも存在する。本質的にポインティングデバイスとも無関係である。遡れば、Excel、Multiplan (マイクロソフト)、Lotus 1-2-3、VisiCalc (ビジコブ) となるが、VisiCalc の開発者はダン・ブリックリン⁵とボブ・フランクストン⁶ はメインフレームの International Timesharing Corporation (ITS) で使われていた Business Planning Language (BPL) や Foresight Systems の Foresight から着想を得たらしい。

さて、プログラミング言語としての Excel と検索すると、VBA (Visual Basic for Applications) や Excel の LAMBDA 関数ばかり引っ掛かってしまうが、それらとは全く無関係な事項として、

³ 1992 年 マイクロソフト創業の功績でアメリカ国家技術賞受賞

⁴ 2007 年 Windows の設計・実装の功績でアメリカ国家技術賞受賞

⁵ 1985 年 VisiCalc 開発の功績で ACM ソフトウェアシステム賞受賞

⁶ 1981 年 VisiCalc 開発の功績で ACM グレース・ホッパー賞受賞

スプレッドシートは「2重の for ループ」と解釈することができる。まさに行列の i, j 要素が一つのセルに相当し、IF 系関数も存在するのだから立派なプログラミング言語である。

POSIX シェルが Unix 使いのプログラミング及び操作環境であるなら、適所なアプリケーションを使わず Excel の世界から一步も出ない「Excel のみ使い」にとって Excel は立派な操作環境なのである。

七つ目の事実を、提案する。

Excel のみ使いはとても奇異だが、多少大らかな気持ちで見守るべき、かも知れない。

但し、2重の for ループと言っても意図したコードになっているかは全要素を隈なく確認する必要があるって、そこは欠点と言える。パターンから外れたセルを検索する機能はあるが、不安は尽きない。そこでお勧めするのが Google スプレッドシートの ARRAYFORMULA 関数、Excel のスピル、である。関数の返り値が配列や行列となる場合に、セル一つの記述で行列を埋めることができる画期的な機能である。

ミニコンを引き合いに出して Excel を取り上げたのは、少々こじつけが過ぎたかもしれない。

ポチクリ文化の継承者たち

以上の歴史を踏まえた上で、Macintosh/Windows 95/XP 文化の「ポチクリ」ばかりの非効率な作業に疑問を持たない、特に計算機科学に携わる読み手に告ぐ。

八つ目の事実を、声を大きくして提案したい。

- ・「ポチクリ」しているオペレータは仕事が極めて遅い、出来ない。
- ・研鑽に努めていないので、説明しても全く伝わらないか、勘違いをする。そして、間違う。
- ・そして、再現性がないので伝達性が乏しい。時間が掛かっている割に成果も乏しい。
- ・しかも、独り善がりな成果しか達成できず、組織として後陣に継承できない。
- ・記録を残そうともしない。口頭で済ませようとする。何故か電話 (LINE 通話) ばかり使う。
- ・国際規格やポータビリティを考慮しない。自分の手元で動作すればそれで終わり。
- ・誤植を気にしない、全角スペースと半角スペースも気にしない。互換「全角ラテン」を使う。
- ・適切な引用が出来ない。引用さえしない。無断掲載する。悪気なく盗用する。
- ・一次情報源にあたらない、近づこうとさえしない。他者の成果の伝達の連綿の意義に無頓着。
- ・ネット検索に頼るのに、使いこなせない、見つからず、思考停止する。不可能と決めつける。
- ・社会の雰囲気、例えば「いらすとや」を安易に使う。Youtube 動画が一般的だと勘違い。
- ・言い訳をする。自分に嘘をついているのに気付かず、正論のように強弁する。
- ・ショートタームとロングタームを区別した的確な未来予測と判断が出来ない。
- ・自分が大事。全体のメリット > 自分のメリット、が実は吉という考え方ができない。
- ・国際化に配慮しない。「\」を「¥」と記述する⁷。管理者なのに JIS キーボードしか使えない。

⁷ 実は Windows 11 になると「\」と「¥」を積極的に区別するようになったが、これまで通り見た目は「フォント依存」なので、初心者は相当混乱するものと思われる。Mac では「\」が「¥」に見えるようなフォントはない。

ここで、Macintosh/Windows 95/XP 由来の「ポチクリ文化」とは手前勝手に怠惰な顧客と養生した「センスや文化とは無縁のマイクロソフト謹製」の、上記を育てた文化と言わざるを得ない。Windows 使いの言い分もあるだろうが、それは参考文献 [3] に譲りたい。

しかし、興味深い例を示そう。同著者の参考文献 [4] にて `sed`, `grep`, `uniq`, `cut`, `split`, `nawk`, `gawk` のようなテキストフィルタコマンドを GUI デスクトップ環境のカットバッファに対して手続的に適用する GUI アプリケーションが紹介されている。これは、確かに便利な場面はあるだろうし、興味深いが、シェルスクリプトや `makefile` として記録に残すだけで済むことを、大仰にも GUI アプリケーションまで開発して、ご自分で使っているらしい。GUI 開発の習作や初学者向けのパイプ (ダグラス・マキルロイ⁸による発明) の啓蒙にはよいかも知れないが、このセンスが参考文献 [3] では触れられていない論点なのだと思う (と言いつつも、少しこの macOS 版を制作したくなった)。

/home/\$LOGNAME — NFS 文化の継承者

NFS (Network File System) とは BSD Unix 及び Sun Solaris で著名なビル・ジョイの成果の一つで、Sun RPC (ONC RPC) の技術基盤によるネットワーク・ファイルシステムである。

マイクロソフトによる SMB/CIFS 共有とはかなり使われ様が異なるが、目的は同じである。

Solaris ではサブネットのすべてのマシンのすべてのユーザのホームが `/home/$LOGNAME` というパスでアクセスすることが出来た。無論、個人情報などは他者に見られないように設定しておくことは各個人の責任であった。

Unix で構築されたサブネットは各マシンはパーソナルであっても、各ユーザーのホームをメンバーに常に顕で、技術と成果の共有がし易い環境だった。そうしたストレージ環境でメンバーが協調してパーソナルコンピュータ・ネットワークを運用する文化が当たり前だったのである。

この文化は、技術的見地から昨今はあまり現実的ではない。何故なら、ローカルストレージの読み書き速度が当時と比べると桁違いに高速だからである。不用意に NFS でユーザのホームをエクスポートしてしまうと、昨今の SSD の高速なパフォーマンスが台無しになってしまう。その辺りのスキルは管理者の腕の見せ所で、本論の範囲を越えるので他書に譲る。

さて、ネットワークファイルシステムに必要なのが、以下のディレクトリ管理システムである。

- ・ NIS … レガシー Unix のディレクトリサービス
- ・ NetInfo … レガシー NextSTEP のディレクトリサービス
- ・ OpenLDAP … LDAPv3 (Lightweight Directory Access Protocol v. 3) サービス
- ・ Active Directory … マイクロソフト Windows Server が提供するディレクトリサービス

⁸ 1995 年 Unix とプログラミング言語の功績で米国科学振興協会のフェローに選出

- ・ Apple LDAP … LDAPv3, Active Directory 対応のディレクトリサービス (NIS は廃止)
- ・ Samba … オープンソースの Active Directory ドメインコントローラ

Active Directory はプロプライエタリゆえ Samba や OpenLDAP の方が好ましい。NetInfo は Apple によって廃止されている。NIS はお手軽だが昨今ではお勧めできない。以上、RPC, NIS, NFS 文化についての参考にされたい。筆者は、例として、文字エンコーディング操作を顧客の需要に応じてか、おかしな箇所でおかしな施しをするマイクロソフト製品を信頼していないので、Active Directory はお勧めしない。全責任を取って将来的に正しいことをする文化が重要である。

Appendix

本論の用語の定義をしておく。

- ・ **Unix** … UNIX と書くとき商標になるので、世間で認められているキャメルケースでこの文化を表している。製品名については UNIX と明記している。
- ・ **mac** … Mac とするとハードウェアとしての Apple 製品も含まれてしまうので、NeXTSTEP 由来の Mac OS X, OS X, macOS と名付けられたオペレーティングシステムとその環境を指す。
- ・ **Mac** … macOS とは呼ばれる以前の Mac OS X 以降のオペレーティングシステムを指す。
- ・ **Macintosh** … Mac OS X 以前を指す。ハードウェアとしては重複する曖昧な時期あり。
- ・ **Windows** … これは明記していなければ、初版 MS-Windows から現在までのすべてを指す。
- ・ **Windows 95** … MS-DOS 時代頃の MS-Windows を指す。
- ・ **Windows NT** … MS-DOS 次世代の DEC 文化を継承した OS を指す。
- ・ **Windows XP** … Windows NT が商業的な成功を収めた象徴とした OS を指す。
- ・ **カットバッファ** … クリップボード、ペーストボード、OS によって呼び名は異なる似た機構。

参考文献

1. 柴田 文彦, 「68000 と Macintosh Toolbox 詳解 — アセンブリ言語プログラミングの奥義」 Rutles, 2022.
… 第4章で ジェフ・ラスキンと Macintosh と Canon Cat について紹介。
2. 伊藤 明雄, 「れがしーなコンピュータ PDP-11: UNIX が開発されたシステム」, Amazon Kindle, 2017.
… 導入でデニス・リッチーとケン・トンプソンが UNIX を DEC PDP-11 に移植を行っている様子や、DEC PDP-11 の開発者デイヴィッド・カトラーについて紹介。
3. 川俣 晶, 「あなたの知らない Windows の1980年代史 ～この世界は嘘だらけ～」, ピーデー, 2021.
… 元マイクロソフト KK 社員による Windows 対 Mac, Unix 等のルサンチマン集。
4. 川俣 晶, 「1995 Linux 対 FreeBSD 戦争勃発・僕のパソコンで UNIX が走るまで」, ピーデー, 2021.
… 「これが超次元エディタだ・Inazuma Procedural Text Editor の開発」の章にて、CUI を GUI で実現！