ヒューマンインタフェース 37－5
（1991. 7. 4）

ビジュアルインタフェースとテキストインタフェースを統合したシェルの設計

宮島靖，河又恒久，小松徹，並木美太郎，高橋延匡
（東京農工大学 工学研究科 情報工学講座）

OS とユーザの間のインタフェースをとるシェルは、主に文字によるインタフェースをとるテキストシェルとアイコンなどのグラフィックスを用いるビジュアルシェルに大別される。しかし、どちらのシェルも一長一短があり、ユーザの操作内容やユーザの熟練度などによって使い勝手が異なる。そこで、両シェルを統合したシェルを設計し、それらを連動するための仕掛けとして内部表現を定め、一か所で管理するようにした。また、インタフェース部をシェルカーネルから分離したため異なるインタフェースのシェルをのせることも容易である。

A Design of the Integrated Shell of Visual and Textual interface

Yasushi Miyajima, Tsunehisa Kawamata, Tooru Komatsu
Mitaro Namiki, Nobumasa Takahashi

Department of Computer Science,
Tokyo University of Agriculture and Technology,

Koganei-shi, Tokyo, 184 Japan

A 'shell' which provides interface between an operating system and a user can be classified into two categories, a textual shell and a visualized shell through graphics. But both shells have advantages and disadvantages. Moreover, their usefulness depends on the kind of user's operation and user's level. Therefore we have designed an integrated shell of both visual and textual interface, and determined the internal representation, which is managed by the common shell kernel for the synchronization of the both interface. Separation among modules of user interface and the shell kernel enables us easily to implement another type of shell interface.

1．はじめに

OS とユーザとの間のインタフェースをとるプログラムとしてシェル[1]がある。初期の計算機では、入力はキーボードによって、出力はキャラクタディスプレイによって行っていた。この時代において、ユーザと計算機を結ぶインタフェース手段は“文字”であり、必然的にシェルもテキスト文字を解釈実行し、結果を文字で表現するテキストシェルであった。その後、マウスやタブレットのようなポインティングデバイス、ビットマップディスプレイなどの出現により、グラフィックスで表示された画面上のアイコンを直接ポインティングデバイスを用いて操作を行うインタフェースのビジュアルシェルが登場した。ビジュアルシェルはアイコンによる直観的な操作とポインティングデバイスによる直接的な操作の新しいインタフェースとして注目された。しかし、現在すべてのシェルがビジュアルシェルになってはおらず、テキストシェルとビジュアルシェルの両方が存在するというのが現状である。このことは、どちらのシェルにも一長一短があり、すべての場合において一方のシェルが優れているわけではないことを示している。

われわれはこの点に着目し、独自に開発したウィンドウシステム“未(HITSUJI)”を利用して、ビジュアルとテキストを統合したシェルを設計した。本論文では、統合のための仕掛けを中心に述べる。

2．それぞれのシェルインタフェースの特徴

われわれは実際に既存のいくつかのシェルを使用してきたが、その経験から感じたテキストシェルとビジュアルシェルのそれぞれの利点と欠点を表１にまとめた。なお、われわれが実際に使用したことのある既存のシェルは次のものである。

・テキストシェル

(1) csh (UNIX, Sun-3)[2]

(2) COMMAND.COM (MS-DOS, PC-9801)[3]

・ビジュアルシェル

(1) Finder (Mac OS, Macintosh)[4]

(2) SX-shell (Human68k, X68000)[5]

表１　ビジュアルシェルとテキストシェルの利点と欠点

	ビジュアルシェル	テキストシェル
利点	・長い名前や名前を知らないファイルを扱うのが楽 ・任意の複数ファイルに対する操作が可能 ・深い階層のファイルを扱うのが楽	・打ち込みやすい簡単なファイル名はすばやく入力できる ・キーボードから手を放さなくて済む
欠点	・ポインティングデバイスに手をのばすのが面倒 ・アイコン指定のためにマウスをかなり動かさなければならない場合がある ・コマンド起動時の引数の指定が面倒	・長い名前や名前を知らないファイルは入力が面倒 ・任意の複数ファイルに対する操作が一度にできない ・深い階層間を行き来するのが面倒

２．１　テキストシェル

テキストシェルでは、ユーザの操作はキーボードから文字列の入力を行うことによって行われる。したがって、操作に必要なコマンド名やファイル名などは前もって知っていなくてはならない。ファイル名に関しては通常ディレクトリの一覧を表示するなどして確認するが、数多くのファイルの中から特定のファイルを探すためには、一つ一つ文字を読んでいかなければならない。

また、キーボードによる文字列の入力も煩わしい。特に、長いコマンド名やファイル名を入力することは煩わしい。キーボードに不慣れな初心者ユーザにとって、キーボードのキー配列に四苦八苦しながら入力することは、本来行おうとしていた作業の進行を妨げかねない。

しかし、そのシステムに慣れており、必要なコマンド名を覚えていて、キーボードもブラインドタッチができるレベルのユーザにとっては、逆にキーボードから手を離さなくてすむテキストシェルはかえって素早い操作ができる場合がある。特にファイル名が短い場合には、キーストローク数が減るので、ユーザの労力は軽くてすむ。

複数のファイルの指定を行うためにワイルドカードが用意されているが、任意の複数ファイルをワイルドカードにマッチングさせることはできない場合が多い。

２．２　ビジュアルシェル

ビジュアルシェルのように、アイコンとマウスを用いたインタフェースは、直観的な操作が可能であり、コマンド名やファイル名を知らなくても操作が可能である。これはユーザがそのシステムに不慣れな初心者である場合や、ファイル名を忘れてしまった場合には便利である。また、ファイルメンテナンスを行う場合などには、一目でファイルの状況が把握できるため便利である。

テキストシェルと違い、任意の複数ファイルを選択することも簡単である。複数のアイコンをポインティングデバイスで直接選択すればよい。

しかし、ウィンドウの枚数が多くなると、ウィンドウの下に隠れる部分が多くなるため、隠れている部分のアイコンをポインティングするためには、ウィンドウの重なり順を変えるなどの余計な手間がかかる。また、キーボードでプログラミングを行っている場合などは、マウスまで手をのばす行為が煩わしい。

２．３　本質的問題

ヒューマンインタフェースの観点から見て、果してテキストシェルとビジュアルシェルでは、どちらの方が使い勝手がよいかという問題がある。われわれは両シェルの使用経験から、それはユーザの熟練度と操作内容、さらにそのときの状況によって異なるという立場をとる。たとえば、短いファイル名を扱う場合にはテキストシェルの方が楽な場合が多いだろう。また、ワイルドカードでは一度に指定できない任意の複数のファイルに対して、削除などの操作を行いたい場合にはビジュアルシェルの方が便利である。さらに、シェル操作を行う直前までキーボードを使用していたかマウスを使用していたかで状況は変わってくる。キーボードを使用してプログラミングを行っている最中では、シェル操作もキーボードで行う方が楽になる場合が多いであろう。つまり、本質的問題はユーザの操作内容などの状況によって使い勝手が変化し、しかも操作内容と使い勝手の間の相関関係が定量的に定められないため、予測ができないというところにある。これに対応するためには、両シェルの間の操作の互換性を徹底的に追求し、それを通して両シェルの評価をする必要があるという結論に達した。

３．シェルの設計方針

われわれが「操作の互換性」に着目したもう一つの理由は、両シェルの長所と短所が表裏の関係にあるからである。すなわち、表１から分かるように、テキストシェルの利点がビジュアルシェルの欠点になり、テキストシェルの欠点がビジュアルシェルでの利点となっている。また、２章で述べたように、どちらのシェルの方が便利かということは、そのときの操作内容やユーザの熟練度によって変わりうる。そこで、これらの性質に着目して、次の設計方針をたてた。

(1) テキストシェルとビジュアルシェルの両方を動作させる

両シェルの欠点をカバーしながら、操作内容やユーザの熟練度に対して柔軟なインタフェースを提供するために、テキストとビジュアルの両方のシェルを同時に動作させることを考えた（図１）。ただし、どちらのシェルを使って操作を行うかは、そのときのユーザの判断にまかせる。

(2) テキストシェルとビジュアルシェルの処理を連動する

両方のシェルを同時に動作させるとなると、ユーザの操作の結果が両方のシェルに反映される必要がある。たとえば、テキストシェルでファイルのコピーを行った場合、コピー先のディレクトリウィンドウ内に、新しく作られたファイルがアイコンとして表示されなくてはならない。

(3) 両方のシェルに共通の内部表現を定める

両方のシェルを連動させるために、操作の内容を両方のシェルに共通の内部表現で表わし、一か所で管理する。それぞれのシェルはその内部表現にあわせて、それぞれのシェルの形態でユーザに反映する。テキストシェルならば文字で、ビジュアルシェルならばアイコンなどでユーザに操作の結果を知らせることになる。つまり、テキストシェルとビジュアルシェルでは、ユーザとのインタフェースのとり方は異なるが、それぞれの操作内容についての OS を駆動する手順は共通である場合が多いので、その共通の部分をシェルカーネルとして一つにまとめて、それぞれのシェルのユーザインタフェースをとる部分を別々に用意するという発想である。

このような構造にしたため、各種インタフェースのシェルの構築が容易になる。他のインタフェースのシェルをのせる場合には、ユーザからの入力を内部表現になおす部分と、内部表現を解釈してそのシェルの表現形態でユーザに結果を反映する部分を作ればよい。これによって、さまざまなインタフェースのシェルを作成して、ユーザインタフェース研究のための実験を行うことが可能である。

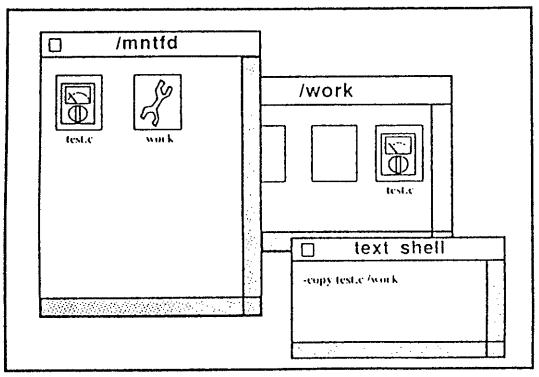

図１　ビジュアルとテキストを統合したシェルのイメージ

４．シェルの設計

４．１　操作対象と内容

本シェルでの操作対象は現段階ではファイルである。また、ファイルに対して次の操作を行うことが可能である。

・ディレクトリを開く
・ディレクトリの作成
・ファイルの移動
・ファイルのコピー
・ファイルの消去
・ファイルの名前変更
・ファイルの実行

４．２　全体構成

シェルの全体構成は図２のようになる。本シェルでは、シェル操作の内容を内部表現で管理する。その内部表現をもとにテキストシェルとビジュアルシェルが、それぞれの表現形態をとるため、このような構成とした。この構成の特徴として次の点が挙げられる。

(1) 内部表現を用いて、複数のシェルでの操作を一か所で管理する

複数のシェルを同時に動作させることを考えた場合、ユーザインタフェース部は各シェルごとに作成するしかないが、各シェルでの OS の機能を駆動する部分は同一のはずである。したがって、OS を駆動する部分は、シェルごとに作成するのではなく、シェルカーネルとして一つだけ用意する。

また、そのためにシェルの種類に依存しない共通の内部表現を用意し、シェルカーネルはそれを解釈して OS を駆動する。

(2) シェルカーネルとユーザインタフェースをとる部分を分離する

既存のシェルで用意されているインタフェースは、テキストあるいはビジュアルのどちらか一方だけであり、これらが同時に動作することは考えられていないため、シェルのカーネルとユーザインタフェースをとる部分は分離されていない。

本シェルでは、実際に OS の機能を呼び出す部分と、ユーザインタフェースをとる部分を分離しているため、ユーザインタフェースの異なるシェルを複数動作させることが可能となる。

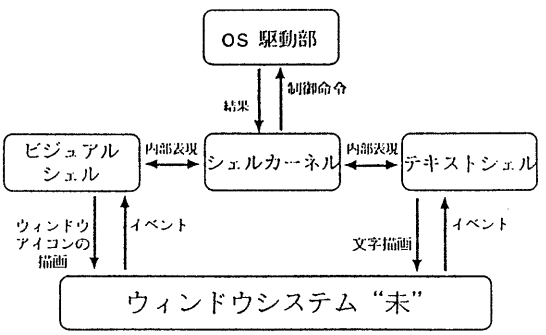

図２　全体構成

次に、各部分の働きを述べる。

(1) シェルカーネル

各シェルから送られてくる内部表現コードを解釈して、実際に OS のファイルシステムやタスクコントロールシステムを呼び出す。そして、すべてのシェル（内部表現を送ったシェル自身も含む）に、送られてきた内部表現に SVC(SuperVisor Call) の結果（エラーコードなど）を付加したものを送る。この仕掛けによって、複数のシェルの連動が可能になる。

(2) OS 駆動部

実際に OS の SVC を発行する部分である。

(3) テキストシェル

テキストシェルは入力部と出力部の２つの部分から構成される。

・入力部

主にキーボードからの入力を受け取り、その文字列を内部表現を作成し、シェルカーネルに渡す。

・出力部

シェルカーネルから渡される操作の結果にあわせて、文字列としてテキストシェルウィンドウに反映する。

(4) ビジュアルシェル

ビジュアルシェルも同様に、入力部と出力部の２つの部分から構成される。

・入力部

ユーザの入力デバイスによる入力を解釈して内部表現を作成し、シェルカーネルに渡す。

・出力部

シェルカーネルから渡される操作の結果にあわせて、アイコン、ウィンドウなどを用いてビジュアルでそれを反映する。

４．３　内部表現

シェルを連動させるために、内部表現を用意するということは前述したが、ここでは具体的な内部表現の仕様を述べる。

４．３．１　内部表現の設計方針

内部表現を決定するにあたって、次の方針をたてた。

(1) 内部表現はシェルに依存しないものにする

容易に他のインタフェースのシェルをのせられるようにするためである。

(2) 拡張性があるものにする

現在シェルの操作対象として考えているものはファイルであるが、将来ファイル以外の資源をシェルで取り扱う場合には、その操作対象についても内部表現で扱う必要がある。

(3) シェル操作は操作プリミティブを用いて表現する

シェル操作を表現するために、ファイルの移動や削除などの操作内容を直接表現する高機能で特化された命令を用意すると、シェル操作や操作対象資源の種類が増えた場合には、それに応じた命令をアドホックに追加しなければならず、柔軟な拡張ができない。そこで、プリミティブを用意し、そのプリミティブと操作対象の種類によって、操作の意味付けが行われるようにする。これによって、少ないプリミティブで多種の操作を表現することができる。

４．３．２　操作プリミティブ

操作プリミティブには次のものを用意する。

・open

操作対象がディレクトリの場合には、ディレクトリを開くという意味になり、ファイルの場合にはファイルの実行となる。

・close

操作対象がディレクトリの場合には、ディレクトリを閉じるという意味になる。

・create

操作対象がディレクトリの場合には、ディレクトリの作成を意味する。

・delete

操作対象がディレクトリまたはファイルの場合には、ディレクトリまたはファイルの消去を意味する。

・copy

操作対象がファイルの場合には、ファイルのコピーを意味する。文字列とファイルまたはディレクトリの組み合わせで用いられた場合には、ファイルまたはディレクトリの名前変更になる。

これらを用意した理由は、これらのプリミティブはある種類の操作対象用に特化されておらず、操作対象の種類によってプリミティブの具体的な意味付けを行えるからである。また、ここに挙げたプリミティブがあれば、現段階で本シェルが用意しているファイル操作は表現可能である。

４．３．３　内部表現のフォーマット

内部表現の一般的なフォーマットは次のようになる。

操作プリミティブ	操作対象１	操作対象２

操作プリミティブによっては、操作対象を１つしかとらない場合もある。

操作対象は OS の SVC 発行時の資源指定のパラメータとして用いられるため、操作対象の表現は OS の資源の指定方法と同じ形式にするのが妥当である。

本シェルは、OMICRON V3（以下 V3）[6][7] 上で動作することを考えて設計したため、操作対象の指定方法もそれに準ずることにした。多くの OS では、ファイルをオープンするには名前を用いるが、V3 ではファイルだけでなく OS が管理する資源すべてについて、「親fd と名前」を用いてオープン

する。親fdとは、オブジェクトを示す id のことであり、オブジェクトに対して SVC の open を発行したときに得られる値である。オブジェクトはツリー構造として管理されており、新しくオブジェクトをオープンしたい場合には、親fd とその下にあるオブジェクトの名前を指定する。新しく得られたオブジェクト id は、さらにその下のオブジェクトをオープンするための親fd として用いることが可能である。このモデルの利点として次の点が挙げられる。

(1) 木構造の途中から直接資源を指定できる

ファイルを指定する場合でも、ルートからのフルパスを持つ必要がなく、親fd からの相対パスによって指定することが可能である。

(2) 親fd にはその資源の性質が継承されているため、資源の種類の把握が容易である

名前だけによる指定では、その名前で示されるオブジェクトがどのような性質を持つかの特定が困難である。しかし、親fd を用いてオープンを行った資源の fd には親fd の性質が継承されている。

また、このモデルはビジュアルシェルでのオブジェクトの指定方式と整合性がよい。ウィンドウの中のアイコンを直接ポイントすることは、ウィンドウを親fd、アイコンを名前に当てはめて考えることができる。

5．おわりに

そのときの操作内容などの状況に応じて、ユーザが使いやすいと判断した方のインタフェースを選択できるようにするために、テキストとビジュアルのインタフェースを持つシェルを設計した。今後は、それぞれのインタフェースと統合したインタフェースの評価を行う。

また、ビジュアルインタフェースなどは、他にもさまざまなものが考えられる。本シェルは、そのような新しいインタフェースを設計し、実際に試すために役立つと考えている。

参考文献

[1] Jerome Howard Saltzer : Traffic Control in a Multiplexed Computer System, MIT project MAC TR-30, Cambridge, 1966.

[2] Getting Started with Sun OS:Beginner's Guide, Sun Microsystems Inc., 1988

[3] MS-DOS 3.3C ユーザーズリファレンスマニュアル, 日本電気, 1989.

[4] Arthur Naiman:The Macintosh Bible, second edition, Japanese-language version, Gijyutsu Hyouron Inc., pp.99-111, 1990.

[5] SX-WINDOWプログラミング, 吉沢正敏, ソフトバンク, 1991.

[6] 岡野, 横関, 並木, 高橋 : “OS/omicron 第３版の設計と実現”, 情報処理学会 OS 研究会資料, 1989.

[7] 横関, 岡野, 並木, 高橋 : “OS/omicron 第３版ファイルシステムの設計と実現”, 第40回情報処理学会全国大会, 1990.3.