流れて行かないUnix環境†

久野 靖†† 角田博保†††

近年，ビットマップ画面とマウス装置を持ち，Unixオペレーティングシステムを搭載した高機能ワークステーションが多く見られるようになっている．しかし実際にこれらのシステムの使われ方を見ると，利用者は単にビットマップ画面上に複数の端末窓を開き，その上で作業をしているにすぎないことが多いように思われる．そこで筆者らはそのような環境の問題点を分析し，ごく少数のツールを作成することでその欠点を補うことを試みた．これらのツールは既存のUnix環境を置き換えるものではなく，Unixの多数の指令群と有機的に組み合わせて使えるようなものとした．これらのツール群は起動されると端末窓とは独立に窓を開き，そこに表示された情報は利用者が陽に指示するまで消えることなく利用可能である．これらのツール群により非常に多数の窓が作られるため，その有機的な関係を整理し使いやすくすることにも留意した．またこれらのツール群の有効性を調べるための簡単な模擬実験も行ったが，その結果，必要な情報がスクロールして消えて行ってしまうという端末窓の問題点が大幅に改善されたとの感触を得た．

1. はじめに

近年，ビットマップ画面とマウス装置を持ち，Unixオペレーティングシステムを搭載した高機能ワークステーションが多く見られるようになっている．これらのシステムにおいてはビットマップディスプレイを使用したウィンドウシステムの機能を活かすことにより，従来の画面端末を用いたUnix環境に比べてより効率のよいソフトウェア開発が行えることが期待される．

しかし実際にこれらのシステムの使われ方を見ると，ウィンドウシステムは単に利用者に多数の端末画面を提供することだけに使われていることが多いように思われる．そこで筆者らはそのようなウィンドウ環境の問題点を分析し，ごく少数の道具（ツール）を作成することによってその欠点を補うことを試みた．

以下ではまずUnix環境をもとにした多くのウィンドウシステムにみられる問題点について述べ，これらの問題点を解消する方向について論じる．続いて作成したツール群について報告し，その評価の試みについても述べる．

2. Unix上のウィンドウ環境の問題点

筆者らはSunワークステーション上のUnixで稼働するウィンドウシステムである X[1,2]，SunTools[3]，GMW[4),5)] 等を利用しているが，いずれのシステムにおいても最も頻繁に利用されるのは，シェル指令を打ち込むとそれを実行しその結果を表示するようなウィンドウであった．しかしそのような使われ方は，見かたを変えれば単に端末画面を1個または複数個提供しているだけであり，したがってウィンドウシステムの利点を十分に活かしているとは考え難い．実際，このような「端末窓」には次のような問題がある．

（1） 表示内容に入力した指令列と実行結果の区別がなく，またそれらの配置も重要度などに関係なく，実行された順序で並んでいるだけである．

（2） 新しく指令を実行したり表示が起こったりすることにより，画面上にある重要な情報がスクロールして行って（流れて行って）消えてしまう．

（3） 場合によっては，一つの指令の出力だけでも窓に収まらず，流れて行くのを「せき止める」ためにmoreなどの指令を組み合わせる必要がある．

（4） 画面エディタ等の全画面を利用する機能を起動すると，これらの情報は全く見えなくなってしまう．

（5） 窓の大きさは表示される情報が必要とされる大きさで決まるのではなく，あらかじめ決められている．

もちろん，端末窓が複数個あればそれらを使い分けて上記のような問題をいくらかは解消できるわけだが，それだけではビットマップ画面の利点を十分に活かしているとは言い難い．

実際，各ウィンドウシステムには，システムの情報を表示する，手紙を整理して読む，等の機能を持つ専用のツールで，ビットマップ画面を活かしたものも用

---

† No 'More' Unix by YASUSHI KUNO (Department of Information Science, Tokyo Institute of Technology) and HIROYASU KAKUDA (Department of Computer Science, The University of Electro-Communications).

†† 東京工業大学理学部情報科学科
††† 電気通信大学情報工学科

意されている．これらはそれ自身では使いやすいが，あくまでも特定用途向けのツールである．そしてUnix そのものをウィンドウ向けに進化させるためには単にこのような特定用途向けのツールを次々と用意するだけではだめなように思われる．これらの問題に対する筆者らの考えを以下次章で述べる．

3. Unix をウィンドウ向けに進化させるには

前章で述べたような問題点を解消する一つの方法は，端末窓とシェルに頼らなくても済むように Unix のツール群を（例えば Macintosh[6] のように）ウィンドウ向けに作り直すことである．しかし，Unix 環境は膨大な数のツールの複合体であり，それらの有機的な関係を保ったままウィンドウに合わせて再構成することは膨大な労力を要する．またそのようにして作り直した環境は現在の Unix 環境とは大幅に異なるものであり，それが現在の Unix の持つスマートさとバランスをそのまま保った使いやすいものであるかどうかは疑問である．

そこで筆者らは別のやり方として，Unix そのものを再構成するのではなく，Unix に少数のツールを新たに用意することで既存の Unix の指令群を自然にウィンドウ環境になじませ，新しい Unix 環境とすることを考えた．そのような新しい環境の開発は，次のような方針をもとに行った．

（1） シェル指令は Unix の使いやすさのためには欠かせないものであり，これを排除するつもりはない．したがって端末窓も最低一つは残す．

（2） 端末窓の問題点はその大部分が，起動した指令の出力がその端末窓に出てしまうことから来ているので，出力を別の窓に移すように考える．

（3） そのために新しいツール群を開発するが，これらは既存の Unix のツール群にとって代わるためのものではなく，それらとうまく組み合わせて使えるようなものでなければならない．

具体的には，Unix を使用している際に（1）繰り返し実行され，（2）その結果端末窓の表示を消したり，表示がスクロールして行ってしまうような指令を取り上げ，それを別の窓に移すことを考えた．もっと直接的に言えば，「端末窓のスクロールをできるだけなくす」こと，言い換えれば「more でせき止めなくても済む」ことが基本的な方針であった（「流れて行かない Unix」および「No 'more' Unix」という題はここから来ている）．以下，実際にどのようなツールを用意したかについて次章で説明する．

4.「流れて行かない Unix」のツール群

前章で述べたような方針に基づき，X-Window (version 10 release 4) 上で動作するツール群を作成した．これらのツール群は X アクセス用ライブラリ Xlib およびウィンドウツール作成用ライブラリ Sx を利用して C で記述し，SunOS3.4, NEWS OS2.1 および Vax/Unix 4.3 bsd 上で動作している．

これらのツールはすべてそれを起動した端末窓とは独立に画面上に窓を開き，陽に閉じるまでその窓はずっと見える状態にある．これらの窓は，重なりの一番上や下に動かす等の制御がすべて同じマウス操作で行え，それぞれの窓に固有な操作は，それぞれの窓が持っているプルダウンメニューから項目を選択することで行える（図 1）．以下に各ツールの概要を述べる．

4.1 xe ファイル

指定したファイルを参照・編集するための窓を作り出す．要するにエディタであるが，これを単独で呼び出すことは余りなく，次に説明する xls や xx から参照・編集用の窓として作り出されることが多い．

4.2 xls [ディレクトリ]

指定されたディレクトリ（またはカレントディレクトリ）のエントリ名が窓に表示される（図 2）．また，そのエントリの一つを選択してメニューから「open」を選ぶと，そのエントリがディレクトリであればエントリ名表示窓が，通常のファイルであればファイル参照・編集用の窓が開かれる．xls はそれ一つで ls, cd, more, エディタを兼ねているといえる．

4.3 xx 指令 引数…

指定した指令が実行され，出力結果が窓に表示される．その出力行がファイル名と行番号を含んでいる場合，その行を選択してからメニューから「edit」を選択するとポインタが行番号に対応する位置にある状態でそのファイルに対する参照・編集用の窓が開かれる．またメニューから「redo」を選択すると指令が再実行される．xx は結果が消えて欲しくないすべての指令と組み合わせられるが，コンパイルや find, grep -n など指令が出力するファイル名と行番号を必要に応じて参照する場合には特に有効である．

4.4 xvr 指令 引数…

指定した指令が実行され結果が窓に表示されるが，xx と異なり指令は一定時間ごとに自動的に再実行さ

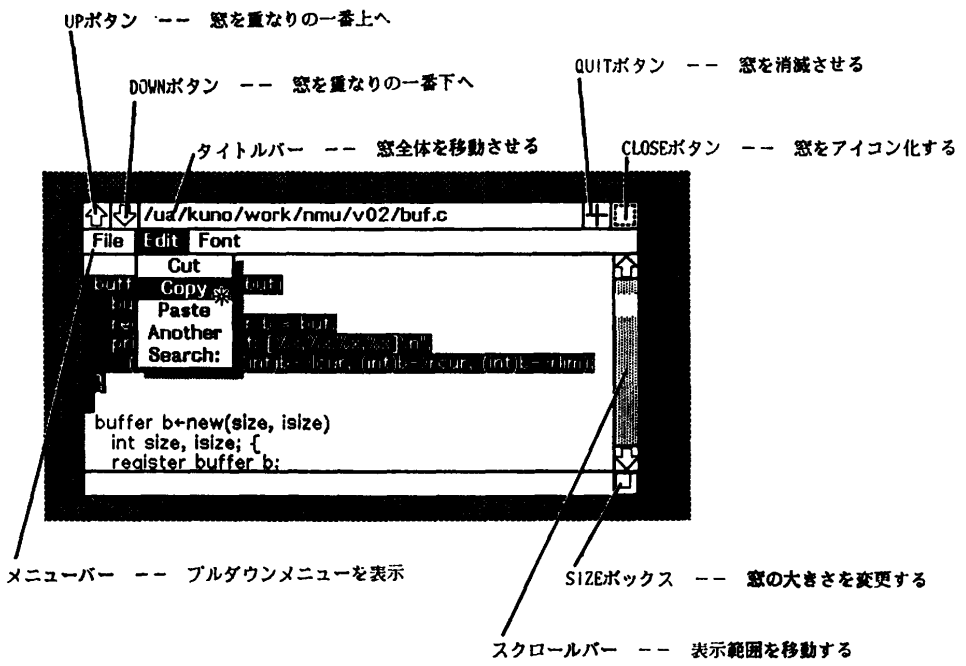

図 1 窓の枠組み (xe の窓の例)
Fig. 1 Framework of the windows (xe example).

れ，表示も更新される．これは ps, rwho などのように絶えず変化する状態を表示する指令と組み合わせて最新の状態を表示させておきたいとき有効である（**図 3**）．

4.5 xt

標準入力から読み込んだものを窓に表示し，同時に標準出力にも書き出す．したがって，パイプラインの中間にはさんでおくことでパイプの中を流れる中間結果をモニタすることができる．

4.6 xcom

起動されると窓を一つ開き，複数の指令行を表示する．表示された指令行の一つを選択し，メニューから「exec」を選ぶとその指令行が実行される．これにより，同じ指令を何回も打ち込まずに済む．これらの指令行はあらかじめ各利用者のホームディレクトリにあるファイルに書いておく．

4.7 xman

起動されると，マニュアル項目の一覧表を表示した窓を一つ開く．その上で項目を選択し，メニューから「open」を選ぶと指定された項目のマニュアルが別の窓として表示され，それを

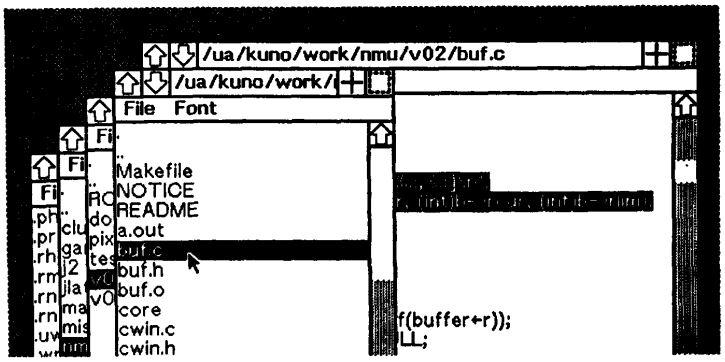

図 2 xls のディレクトリ窓と編集窓
Fig. 2 Directory/edit windows of xls.

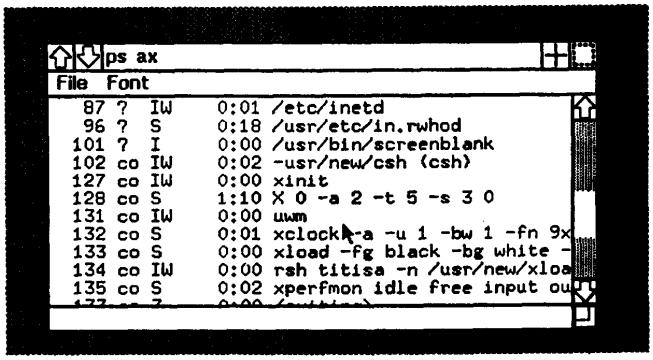

図 3 「xvr ps ax」の例
Fig. 3 How "xvr ps ax" looks like.

読むことができる(図 4). さらに, 特定の項目を探すための機能としてキーワードを指定し, それを表題に含むような項目を探したり, そのような項目のみをふるい分ける機能も用意されている (図 5).

4.8 xw

起動されると, 画面上にある窓の一覧表を表示した窓を開く. その上で項目を選択し, メニューから「open」,「close」を選択することで指定した窓を開いたり閉じたりすること, および「clean」メニューを選択することで開いてあるすべての窓をまとめて閉じ, アイコンの場所を整頓することができる. xw はいわゆるウィンドウマネージャのようなものであり, その仕事は他のツールが作り出す窓を制御することである.

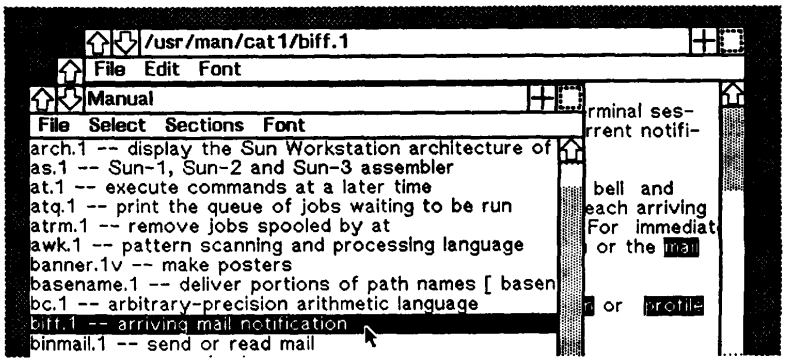

図 4 xman の索引窓
Fig. 4 An index window of xman.

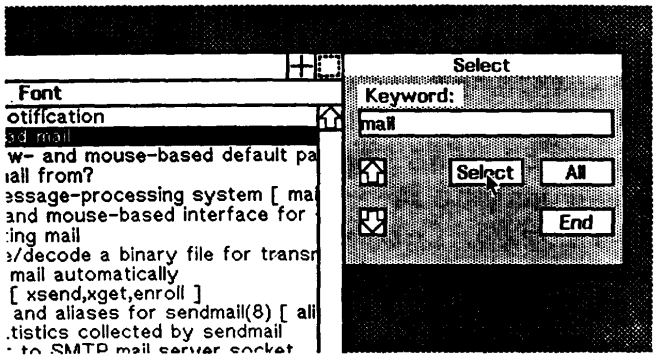

図 5 xman のサーチボックス
Fig. 5 A search box of xman.

前章で述べたように, これらのツールは基本的に端末窓の表示が流れて行ってしまうような場面を考慮し, それに対して必要なツールを設計するという方針で順次作って行ったものである.

例えば xvr は who や ps のようにしばらく経つと状態が変化するため繰り返し実行し, そのため表示が流れて行ってしまうものに対処するため作成したし, man や more はそれまでの端末窓を全部消してしまうためその代わりとして xman や xe を用意した. ls は場合によっては表示量が多いこともあり, ファイルを探すような場面では cd と組み合わせて繰り返し使われるという両方の性質があるため専用の xls を作ったものである. 特に xx は他の指令の状況に関わらずしばらく出力を残しておきたい場合や表示量が多い場合全般に使用するために作られたもので, もっとも汎用性があるツールであるといえる.

これらのツールを使用し始めると, 短時間に非常に多くの窓を作ってしまい, その整理に手を取られるようになったのでこれに対処するため xw のようなものも用意した. これらを含め, 窓の統合化の問題について次章で述べることにする.

5. ツール間の統合化

前章で述べたツール群はそれぞれ独立したプログラムではあるが, 一つの画面で並行して (あるいは有機的に関連させて) 使用するものであるため, それら一つ一つの機能についてだけでなく, 全体として組み合わせたときの使いやすさについても考慮することが重要である. 具体的には各ツールの作る窓にできるだけ共通の枠組みを持たせること, 窓の画面上での配置, 多数の窓の整理などが問題になる. 以下本章ではこれらの点についてどのような工夫をしたかについて論じる.

5.1 窓の枠組みと機能

窓の基本的な機能はツールから利用者に提示する情報を表示し, また利用者がツールに (キー入力やメニューを通じて) 指令を伝える媒介をすることである. しかし, 多数の窓を使いながら作業を行うようになってくると, 窓の配置を変えたりとりあえず開いていなくともよい窓をアイコン化したり等の窓自身の制御のしやすさが環境全体の使い勝手に大きく影響するようになってくる.

X, SunTools, GMW 等 Unix 上のウィンドウシステムでは制御シフトキーまたはウィンドウマネージャメニューとマウスボタンの組合せを使用したウィンド

ウマネージャが多く使われているが，このような方式はあらかじめマニュアル等を読んでおかないと使い方がわからず，直接操作感に欠けるという欠点があるように思われた．そこで本ツール群では文献 6) のように情報を表示する窓本体の周辺に窓を制御する領域を付随させる方式を取った（図1参照）．この方式では窓の様々な制御をする部分がそれぞれ画面上の領域として形を持つため，そこをつついてみればその機能が試せ，一度覚えた機能は絶えず見えているため忘れる心配が少なく，直接操作感があるという利点が大きい．

本ツール群では各ツールの提供する窓が共通した枠組みを持つため，利用者はそれぞれの違いを意識せずに一連の窓を扱うことができる．ツールごとの違いはメニューの内容に反映されているが，利用者はこれをプルダウンメニューを（選択はせずに）降ろしてみることで知ることができる．これらの窓の枠組みは文献 6) に近いものであるが，一つの画面上で多数のツールが並行して走り，したがって一時に非常に多数の窓を扱うこと，窓は一番手前に持ってこなくても操作できること，プルダウンメニューが窓ごとについていることなどからその操作感はかなり異なったものとなっている．

5.2 メニュー内容の共通化とショートカット

上述のようにメニューの内容は基本的に個々のツールごとに異なるが，その場合でもメニューの構成をできる限り揃えている．例えば最左端のメニューはすべて「File」という名前がつけられており，ツール内で新しい窓を開いたり，ツールを終了させる機能は原則としてここに配置している．同時に，「Fonts」という名前のメニューは常に右端にあり，これを選択することで窓本体の表示に用いるフォントを選択できる．

また，多くのツールでは「特によく使われるメニュー機能」が存在し，これらを毎回プルダウンメニューから選ぶのは煩わしい（例えば xls での選択＋open がこれにあたる）．そこでそのような機能についてはマウスでダブルクリックすることにより起動できるなどのショートカットを用意した．特にダブルクリックによるショートカットは各ツールについて統一性を考慮し，「よく使われる自然な機能」に対応させるようにした．これらの構成も文献 6) を参考にしているが，一時に多数のメニューが画面上に共存することから，その共通化はさらに重要であるといえる．

5.3 窓の配置

窓はそのタイトルバーをつかんで動かすことで簡単に移動できるので，窓が作られた後でそれを利用者が好ましい位置に移動することは容易である．しかし，できれば窓がはじめから使いやすい場所に自動的に作られるのに越したことはない．そのため，ツールが起動されて最初の窓を開くときには現在画面上に開かれている窓の場所を調べ，できるだけ空いている場所に開くようになっている．空いている場所が見つからないときはとりあえず左下隅に作られるが，これは他の場所に比べて移動するためにつかむのが楽なように感じたためである．

一つのツールのもとで関連する窓を開く場合には上記の方法ではなく，図2のように最初の窓から少しずつ右上にずらして開くようになっている．これは，画面上の空いている場所は貴重な資源であり，むやみに消費するのは望ましくなく，また関連する窓が近い場所に開くことが使いやすいと考えたためである．右上にずらして行くのは窓の左上隅に UP ボタンがあり，このように配置することで一群の窓のうちどれが手前になってもこのボタンは隠されず，そこを押すことでどの窓でもすぐに手前に出せるようにしたからである．

5.4 アイコンの配置

X などでは一時に作る窓の個数が比較的少ないため，窓を閉じてアイコン化させることも余り多くなく，その配置もそれほど問題ではない．しかし本ツール群の場合には窓が多数作られるため，アイコンの配置も重要である．少なくとも筆者らの経験では，各窓について利用者が自分でアイコンの場所を決めるのでは煩わしくて仕事にならず，ある程度適切な位置に自動的にアイコンが配置されることが最低限必要であった．

現在のところ，本ツール群ではアイコンを画面の右下隅に配置するようにしているが，これはここが開いている窓との干渉が最も少ないと考えたためである．具体的な配置方法としては画面上にアイコンの入る「ます目」を考え，そこに一つずつアイコンを入れることで揃えて配置する．一つのツールが一つの窓しか作らない場合には一番右端の列を選び，既に存在するアイコンのうち一番上にあるもののすぐ上に作る（このようにしても右端の列があふれることはまれであった）．一つのツールが複数の窓を作る場合にはまだアイコンが置かれていないできるだけ右寄りの列を選び，そこに下から順にアイコンを配置する（図 6）．

5.5 窓の大きさとフォント

窓が開くときの大きさについても，窓の配置と同様な問題がある．多くのウィンドウシステムの端末窓は（おそらく歴史的事情から）80文字×24行が収まる大きさを標準値としているが，本ツールでの経験からみるとファイルの内容表示には80文字では広すぎて画面の浪費であるし，行数は逆にもっと多い方が好ましいようである．

この問題に対する一つの回答は，表示する情報に応じて自動的に窓の大きさを決めることのように思われるかも知れないが，実際にはそれほど単純ではない．例えばファイル表示の場合でも他の窓を隠してまで全内容を表示させたいかどうか，全行数を一度に表示できないときは何行分を見せるか，窓の幅はどうするか（最長行の幅を取るのは明らかに望ましくない），編集等によりこれらの値が変化したときは窓の大きさも変化するのか（これも使いやすいかは疑問である），などの問題がある．

加えて，同じ大きさの窓でもフォントによって表示できる情報の量が異なるという要因も考慮する必要がある．例えばテキストファイルではポイント数は同じでも可変幅のフォントを使うことで固定幅のものに比べて多くの文字数を表示できるが，プログラムの出力などを表示するときにはカラムが縦に揃わないと見にくいため，固定幅のフォントを使わざるを得ない場合もある．

本ツール群ではとりあえず，ツールごと，窓の種類ごとに標準の大きさとフォントを用意し，これらの標準値は利用者ごとに設定できることとした．全体的な方針としては，始めは他の窓を隠さないように小さめに作り，必要なら利用者が簡単に大きさやフォントを変えられるようにする，という行き方を取っている．

5.6 窓の整理

本ツール群を使い初めてすぐに，窓が気軽に開けることは大きな利点であるが，一方で多数の窓を整理するのにかなり手間を要することも明らかになった．そこで xw の機能として現在開いている窓をまとめてアイコン化する機能を用意した．このようにしたのは，ツール群を使ってみた経験から，窓が多すぎると感じるのは仕事の切れめで新しい窓を開こうとしたときが多く，したがって既存の窓はほとんど閉じても構わず，残って欲しい2，3の窓について例外扱いする手間よりもすべて閉じた後で必要なものを再度開く方が早いと考えたためである．

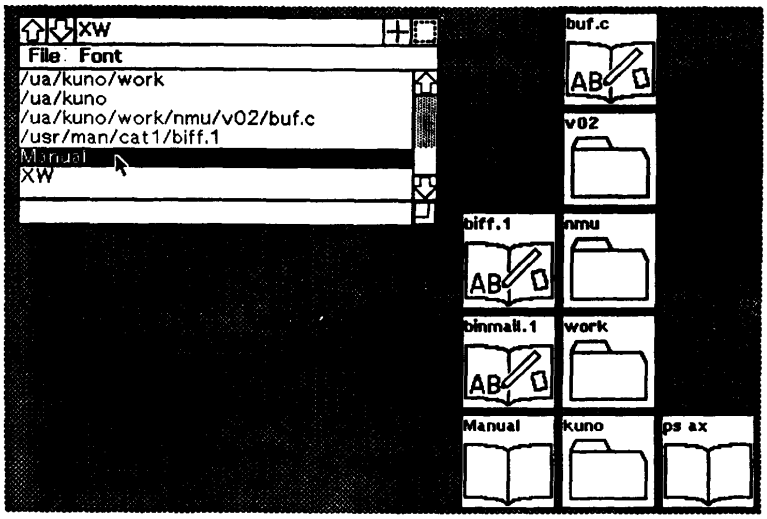

図 6 xw の窓とアイコンの配置
Fig. 6 An xw window and layout of icons.

ただし，時計やコンソールウィンドウなど，ウィンドウシステム開始時に作ってずっと置いてある窓は閉じて欲しくない．周囲の利用者を観察した結果，これらの窓は画面の上または左右端に接して配置することが多いことがわかったので，整理の際にはそのような窓は閉じないように配慮した．

6.「流れて行かない」ツール群の簡単な評価

とりあえずツール群が動くようになった段階で，新ツール群の効果を見るために簡単な評価を試みた．具体的には Unix で行われる典型的な作業（ここでは小さなプログラムの開発）を取り上げ，それを裸のX上で行う場合と新ツールを用いた場合とで利用者の挙動がどのように異なるかを比較する模擬実験を行った．その概要は次のとおり．

- 作業は，ファイル名を引数として与えるとその名前を変更日付と共に打ち出すプログラム fdates を作ることである．
- 被験者は Unix やCは知っているが，システムコールやライブラリについてはマニュアルを見ないとわからないものと仮定する．
- 実際に筆者らが被験者になったつもりで作業を行い，その記録をもとに「典型的な」被験者が行うであろうセッション記録を作成する．

模擬実験の結果を図7に示す．ここでは新ツールの場合を左，Xの場合を右，共通する部分を中央に記している．網掛けはマウスによる操作を示す．かっこ内は説明である．

結果中の連続する網掛けの数を数えるとマウスに手を移動した回数がわかるが，その値は新ツールでは10回，Xでは8回であり，両者の差は予想より少なかった．ただし，Xでは基本的に手はキーボードにあり，使用する窓を選ぶためにマウスに手が動くのに対し，新ツールではふだんはマウスで操作している文字列を打ち込む時にキーボードに手を動かすことになる．

次に模擬実験で感じられた新ツールの利点および欠点について述べる．利点としては，次のものが挙げられる．

- マニュアル等を眺めるのはマウスを使うためやりやすい．
- 特に，関連する項目が多数ある時は窓が自動的に開くことの利点が大きい．
- マニュアルや include ファイルから必要な場所を cut/paste により持って来るのが容易である．
- エラーメッセージから対応する行へいって編集するのが容易である．
- 画面から情報があふれてしまう心配がない．

一方，新ツールの欠点としては次の点が特に大きく感じられた．

- ファイル名の新規指定，キーワードの指定など，新たに文字列を打ち込まなければならない場所でまだるっこしい．

これは，ふだんキーボードに指があればそのような指定は指令と同列に入力できるのに対し，本ツールでは通常マウスに手があるため，それをキーボードに移動する心理的負担が非常に大きいためだと思われる．この問題はマウスを使用する多くのシステムに共通のものであり，簡単に解決できるものではないがとりあえず次のような対応策を考えている．

新ツールの使用	X-Windowの使用
（作業ディレクトリへ行く）	
xls $cwd /usr/include	vi fdates.c
fdates.cを開く	
（プログラムのあらましを作る）	
main(argc, argv)	
int argc; char *argv[]; {	
for(i = 1; i < argc; i++) {	
printf("%s¥n", argv[i]);	
}	
}	
（日付をとる方法が分からず man -k を見ようとする）	
マウス移動して端末窓へ	viを中断する
xman	（新しい窓を作ろうとする）
（xmanの窓ができる）	xterm &
メニューから「Select:」を選ぶ	場所と大きさをマウスで指定
キーワード"date"を打ち込む	（新しい端末窓ができる）
「select」ボタンを選択	man -k date
（単語dateを含む項目を表示するが見つからずfileで再試行）	
キーワード"file"を打ち込む	man -k file
「all」、「select」を選択	（画面に入りきらなかった）
	man -k file | more
（項目を眺め、statだと分かる）	
その項目をダブルクリック	man stat
（内容を読み、includeが必要と分かる）	
マニュアルからincludeの例をコピー	viの窓へ戻る
	viを再開する
	#include <sys/types.h>
	#include <sys/stat.h>
struct stat buf;	
stat(argv[i], &buf);	
（日付がどこに返るか調べ、time_t st_mtime; と分かる）	
（time_tの型が知りたいと思う）	
xlsの窓へ行く	（もう一つ窓が欲しい）
<sys/types.h>を開く	さっきの端末窓へ行く
	xterm &
	場所と大きさをマウスで指定
	more /usr/include/sys/types.h
（time_tを探しlongと分かるが、これを文字列にする関数は？）	
キーワード"time"を打ち込む	man -k time | more
「all」、「select」を選択	
（ctimeだと分かる）	
その項目をダブルクリック	man ctime
（宣言が必要だと分かる）	
宣言をcut&pasteでコピー	viの窓へ行く
	char *ctime();
printf(...)を改訂	
（一応完成、翻訳する）	
メニューの「save」を選択	:w
xx cc -o fdates fdates.c	さっきの端末窓へ行く
（xxの窓ができ、翻訳開始）	cc -o fdates fdates.c
（iが未定義のエラーがでる）	
そのメッセージをダブルクリック	viの窓へ行く
（新しい窓ができ、エラーの行にいる）	
int i; を追加	
「save」を選択	:w
「redo」を選択	さっきの端末窓へ行く
	!!
（翻訳が成功）	
端末窓へ行く	
fdates * を実行してみる	
（成功した。作業おわり）	

図7 模擬実験のサンプルセッション
Fig. 7 Sample session record of the experiment.

- できるだけ新たな文字列を打ち込まずに済むようにする．そのためには例えば「探したそうなキーワードや文字列の集まり」「使いたそうなファイル名の

集まり」などを入れた「箱」を用意して，必要ならそこから取って来れるようにする．

- キーボードに手が移ったら，しばらくはそのままでよいようにする．そのためマウスでできる操作はすべてキーボードでもできるようにしておく．

特に後者については「キーボードから手が動くのは耐えられない」と考えている利用者に本ツール群を使ってもらうためにも重要である．また，そうでない場合でもいくつかの文字列を入れるところで連続してキーボードを使い，それが済んだらマウスに戻れるようになっていればかなりフラストレーションは減らせると思われる．例えば図7左の場合でも「指令 xman を打ち込む」ところから「all, select を選択」までがすべてキーボードで行えればマウスとの間の手の移動が2回少なくなり，ずっと快適であろうと思われる．

以上の問題を除けば全体的な感想としては新ツールの方が直接操作感があって使いやすく思われた．また，今回の作業例ではプログラムが小さくファイルも1個だったこと，一つの作業のみを実行したこと，などから新ツールの有利さは比較的活かされなかったように思えるので，より大きく現実的な作業を対象にした評価も行いたい．

7. まとめ

本ツール群については，現在のところ筆者らによる試用と並行して各種機能の拡張・手直しを行っている段階であり，まだ不完全な部分も多い．しかし，現在でも端末窓がスクロールしていって必要な情報が見えなくなってしまう，というフラストレーションは大幅に減少したことが感じられ，主観的には「より快適な」環境を作り出すことができたと考えている．

今後はより多くの利用者にツールを使用してもらうことで使いにくい部分を発見・改良していくとともに，より厳密な実験に基づいて上記の「主観」の客観的な裏付けを行うことを計画している．

参考文献

1) Scheifler, R. W. and Gettys, J.: The X Window System, *ACM Trans. on Graphics*, Vol. 5, No. 2, pp. 79-109 (1986).
2) Gettys, J. et al.: Xlib—C Language X Interface Protocol Version 10, MIT (1985).
3) Sun Microsystems, Inc.: SunView Programmer's Guide, Mountain View (1986).
4) 萩谷：GMW ウィンドウシステムについて，*bit*, Vol. 19, No. 3, pp. 4-19 (1987).
5) 大谷ほか：複数のウィンドウを組み合わせて一つのアプリケーションを作成する，日経エレクトロニクス，No. 426, pp. 195-209 (1987).
6) Apple Computer Inc.: *Inside Macintosh Volume I-IV*, Addison-Wesley, Reading (1985).
7) 久野，角田：流れて行かない Unix 環境を目指して，究極のプログラミング環境，pp. 81-88, 情報処理学会・プログラミングシンポジウム委員会 (1987).
8) 角田，久野：流れて行かない Unix 環境の評価，第29回プログラミングシンポジウム報告集，pp. 95-104 (1988).

(昭和63年3月14日受付)
(昭和63年6月24日採録)

久野　靖（正会員）

1956年生．1981年東京工業大学大学院理工学研究科情報科学専攻修士課程修了．1984年同専攻博士後期課程単位取得退学．同年東京工業大学理学部情報科学科助手．理学博士．プログラム言語，オペレーティングシステム，プログラミング環境，分散システム等に興味を持つ．ACM，日本ソフトウェア科学会各会員．

角田　博保（正会員）

昭和25年生．同49年東京工業大学理学部情報科学科卒業．同51年同大学院博士課程修了．同57年同大学院博士課程修了．同年電気通信大学計算機科学科助手．理学博士．文字列処理，プログラミング方法論，ヒューマンインタフェース等に興味を持つ．ACM，日本ソフトウェア科学会各会員．