sun®
microsystems

Windows *and* Window Based Tools: Beginner's Guide

Part No: 800-1287-03
Revision A of 17 February 1986

Credits and Trademarks

Contents

Tables

Figures

Preface

This document introduces SunView, the Sun Visual/Integrated Environment for workstations, that provides an advanced user interface, high-resolution graphics, and SunWindows, a window system, which all run on the Sun workstation. We assume that you have some experience with the Sun Workstation, and the UNIX[†] operating system.

We provide examples to learn the system, not detailed explanations of the inner workings of the window system. However, as in each of the *Beginner's Guides*, we refer to the other Sun documentation, drawing a road map for you to follow when you wish to learn more about a certain topic.

Windows and Window-Based Tools describes how to set up the window system, create and manipulate windows, and store window attributes. In addition, this manual presents examples of window-based tools, and includes special chapters that describe advanced window system topics, other useful features, a glossary, and a quick reference.

Prerequisite documents

Getting Started With UNIX: Beginner's Guide

Companion documents

Setting Up Your UNIX Environment: Beginner's Guide
Self Help With UNIX: Beginner's Guide
Windows and Window-Based Tools: Beginner's Guide
Mail and Messages: Beginner's Guide
Games, Demos, and Other Pursuits: Beginner's Guide
Doing More With UNIX: Beginner's Guide
Using the Network: Beginner's Guide

Formatting Documents on the Sun Workstation
Editing Text Files on the Sun Workstation
Commands Reference Manual

† UNIX is a trademark of AT&T Bell Laboratories.

1

The Window System

1

The Window System

A *window system* allows you to divide your screen into work areas, in which you can run programs, edit files, and send mail messages, just as you do when the screen has only one work area.

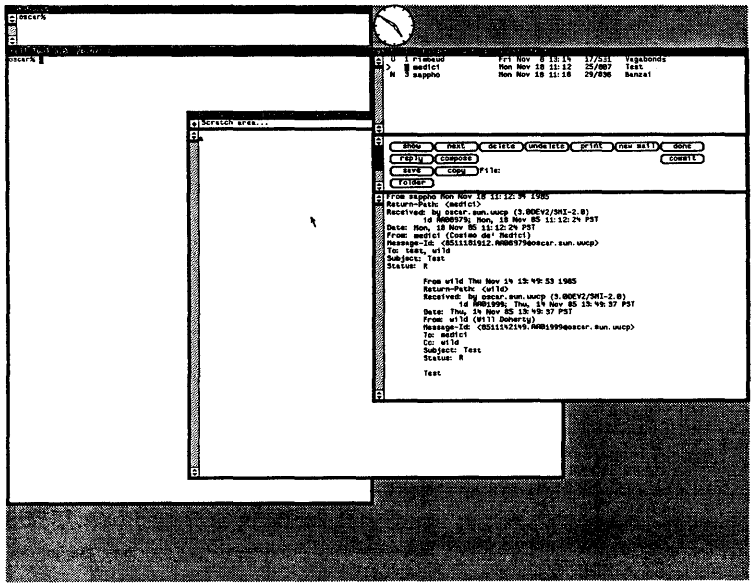
Imagine that your workstation screen is a desktop, and the work areas on the screen are sheets of paper that you work on, or stack in piles, as you wish.

With the window system, you can accomplish many tasks simultaneously by working on each task within its own work area. This chapter describes what the window system looks like, the relevant parts of the Sun Workstation, and the difference between a *cursor* and a *caret* within the window system.

1.1. A Window System in Operation

Your workstation may look something like this after you log in:

Figure 1-1 *A Window System in Operation*



```
oscar%
```

```
oscar%
```

```
 U   1 rimbaud          Fri Nov  8 13:14    17/531    Vagabonds
>    2 medici           Mon Nov 18 11:12    25/887    Test
 N   3 sappho           Mon Nov 18 11:18    29/898    Banzai
```

```
From sappho Mon Nov 18 11:12:54 1985
Return-Path: <medici>
Received: by oscar.sun.uucp (3.0DEV2/SMI-2.0)
        id AA06979; Mon, 18 Nov 85 11:12:24 PST
Date: Mon, 18 Nov 85 11:12:24 PST
From: medici (Cosimo de' Medici)
Message-Id: <8511181912.AA06979@oscar.sun.uucp>
To: test, wild
Subject: Test
Status: R

        From wild Thu Nov 14 13:49:53 1985
        Return-Path: <wild>
        Received: by oscar.sun.uucp (3.0DEV2/SMI-2.0)
                id AA01999; Thu, 14 Nov 85 13:49:37 PST
        Date: Thu, 14 Nov 85 13:49:37 PST
        From: wild (Will Doherty)
        Message-Id: <8511142149.AA01999@oscar.sun.uucp>
        To: medici
        Cc: wild
        Subject: Test
        Status: R

        Test
```

When your screen looks like the previous figure, a window system is running on your workstation. Each of the rectangular screen sections, or work areas, is a *window*.[1]

Note: A *bitmap* screen allows the programmer to access individual *pixels*, or dots on the screen, to draw text and pictures.

When your screen doesn't look like the previous figure, you can run the window system on your workstation (you'll learn how in Chapter 2 , on setting up windows). The window system will not run on a simple terminal; it requires a workstation with all of its parts, and a *bitmap* screen.

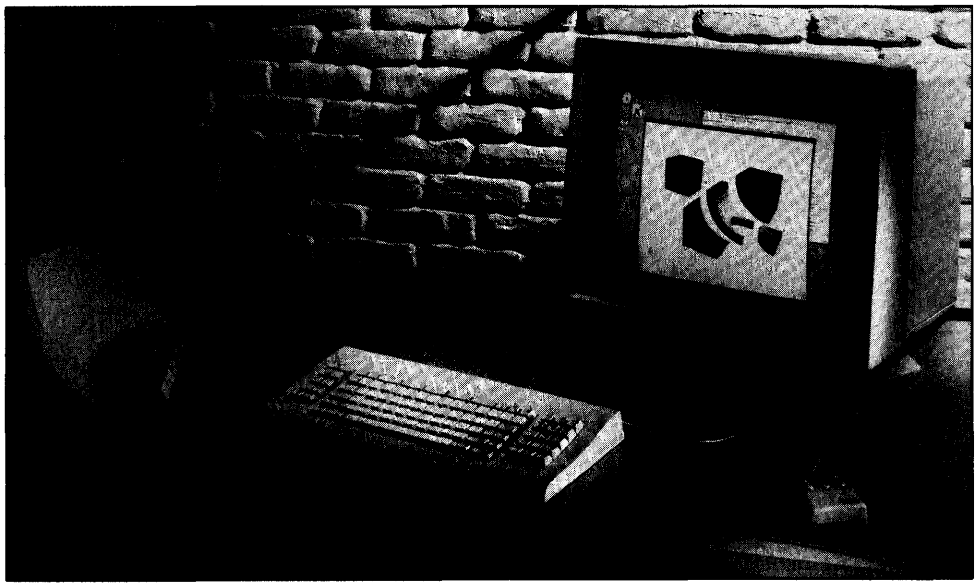
1.2. Parts of the Sun Workstation

To use the window system, you need to know the following parts of the Sun Workstation:

- the *screen*
- the *mouse*
- the *mouse tablet*
- the *keyboard*

If you don't remember these parts, check the chapter on the terminal screen and mouse in the *Getting Started with UNIX: Beginner's Guide*.

Figure 1-2 *Parts of the Sun Workstation*



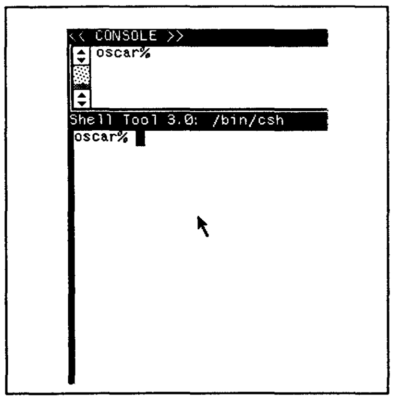
[1] The screen illustrations in this manual were made on a system with an 11-point default font. If your system administrator chose a different default font, your screen won't correspond exactly to the manual's screen illustrations. However, if you want to set your default font size to that used for the manual, you can do so using `defaultsedit`. The `SunView` category has a `Font` attribute that you should set to `/usr/lib/fonts/fixedwidthfonts/screen.r.11`. For more information, see the section on `defaultsedit`, Section 11.1 .

1.3. Cursor Versus Caret: Window System Locators

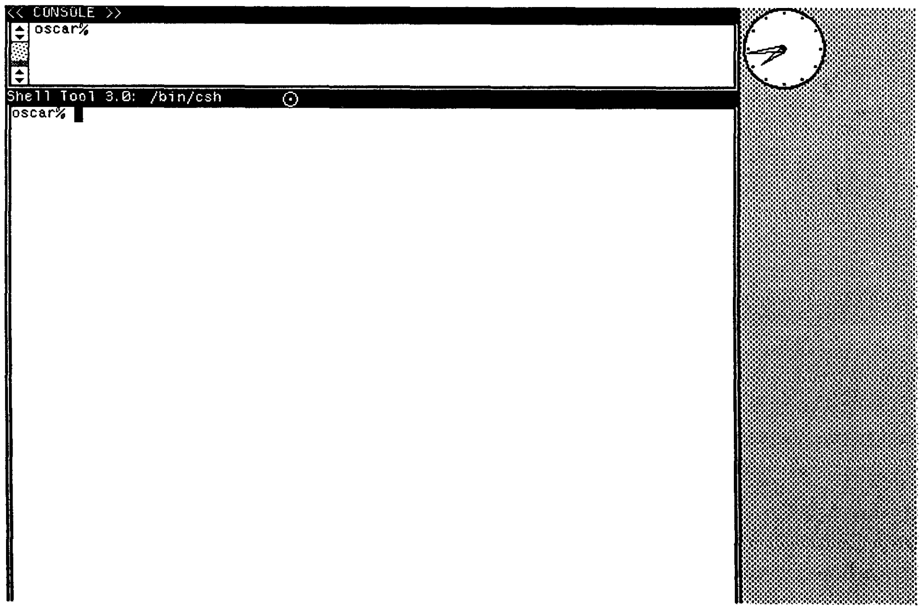
You also must understand the distinction between the *cursor* and the *caret*. When we talk about the window system, the definition for *cursor* is unlike the definition we have used so far. In window system terminology, the cursor is not the rectangular section of screen that tracks your current type-in location on the screen. Instead, the window system cursor is an arrow, or sometimes a target circle, that tracks the movements of the mouse.[2] The cursor enables the mouse as a *locator* device with which you can pick, or designate, a spot on the screen.

The *caret*, then, is a small rectangular, triangular, or diamond-shaped section of screen that tracks your current type-in location on the screen. In some window applications, or *tools*, the caret blinks on and off or becomes only an outline of its former shape, a "shadow" caret.

Figure 1-3 *Cursor v. Caret: Window System Locators*



Enlarged view



In this chapter, you learned what a window system is and what it looks like. You reviewed the parts of the Sun Workstation and discovered the difference between the cursor and the caret within the window system.

[2] It is possible to encounter cursors of different shapes in some window-based tools.

2

Setting Up Windows

2

Setting Up Windows

In this chapter, you will learn how to:

- Start `suntools`
- Fix ''stuck'' mice
- Use the mouse to open an icon
- Start `suntools` automatically upon login

2.1. Starting `suntools`

To start the window system, log in to your machine, then type `suntools` to the UNIX† command prompt.[3]

Figure 2-1 *Starting* `suntools`

```
venus% suntools
```

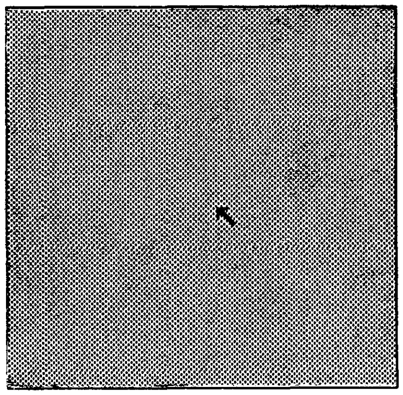
At this point, the screen clears, then a granular gray surface appears.[4]

† UNIX is a trademark of AT&T Bell Laboratories.

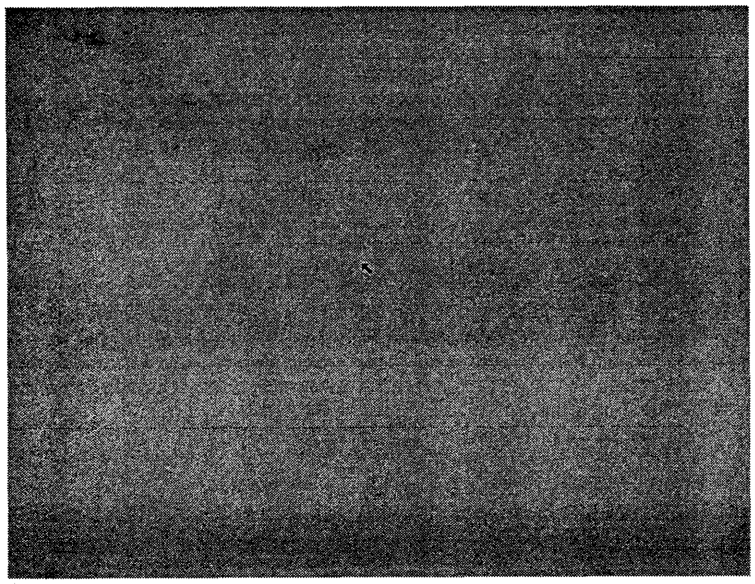
[3] The command prompt is `venus%` in this case, because the example machine is called venus.

[4] If you get an error message, you may be trying to run the window system on the wrong equipment. Contact someone who is using the window system or your system administrator.

Figure 2-2 *Granular Gray Surface With Cursor*



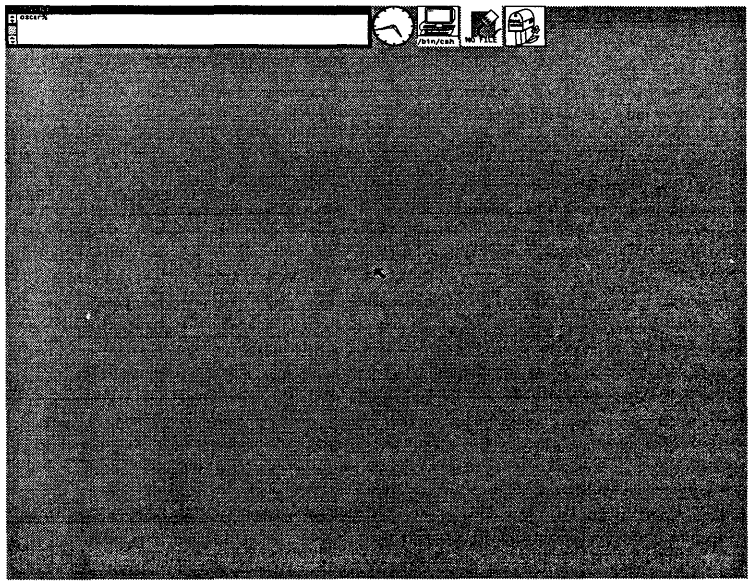
Enlarged view



Note: The *console* window is a special window that displays messages about the status of the system.

Soon, objects appear along the top of the gray background screen. Most of the objects are *icons*, or "closed" windows, except for one window, the *console* window, that is open. An icon is a little drawing that represents a closed window. You must open an iconic window before you can type commands to the window. `suntools` is ready for your instructions when the screen looks like this:

Figure 2-3 `suntools` *Screen: Ready for Instructions*

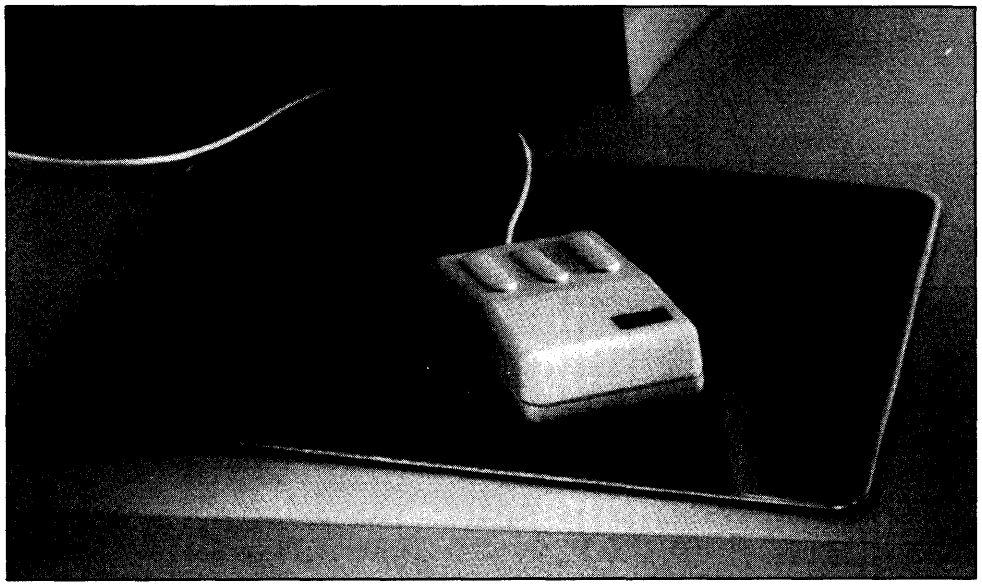


The amount of time it takes for `suntools` to start depends on the number of tools you want it to initiate and display on the screen.[5]

Though it is inconspicuous at first glance, look for the black arrow somewhere on the gray screen, usually in center screen pointing ''northwest.'' That is the cursor.

To move the cursor, push the mouse about on its tablet. Make sure that the mouse's ''eyes'' are on its underbelly, in other words, the mouse buttons are on top; then, the mouse can read the lines on the tablet to indicate relative position on the screen. Also, make sure to orient the mouse perpendicular to one of the long sides of the tablet, so that the cursor responds appropriately when you move the mouse — if the cursor doesn't respond properly, rotate the mouse on the tablet, then try moving the cursor again.

Figure 2-4 *Moving a Mouse on Its Tablet*



2.2. ''Stuck'' Mice: No Cursor Movement

If the cursor won't move when you push your mouse on its tablet, your mouse may be ''stuck.'' Try pushing the mouse rapidly in large circles on the tablet several times. The mouse should then start to control the cursor position.[6]

[5] See the chapter on storing window attributes to learn about controlling the number of windows on the screen just after starting `suntools`.

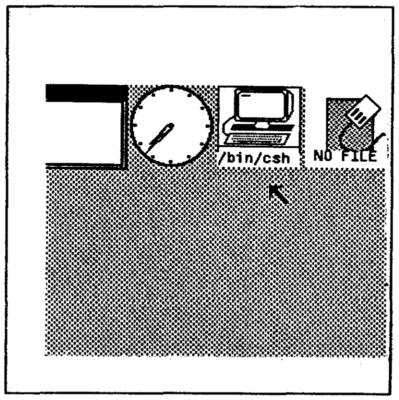
[6] If the mouse still won't control the cursor position, check to make sure it is plugged in to your workstation without a loose connection. If that fails to solve the problem, unplug the mouse from the workstation and plug it in again. If it still doesn't work, the mouse or the workstation may be defective — contact your system administrator.

2.3. Open a Window

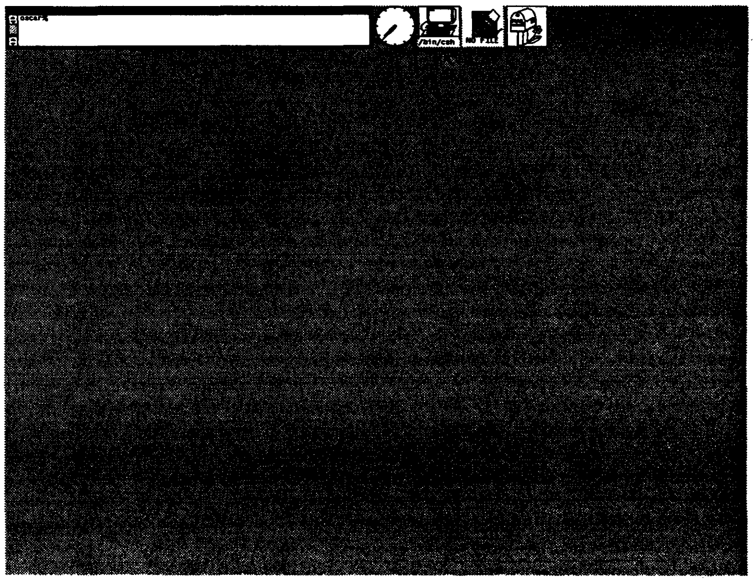
There it is — the `suntools` screen. What useful things can you do with it?

Push the mouse so as to move the cursor onto the small rectangular icon that looks like this, the *shelltool* icon:

Figure 2-5 *Shelltool Window Icon*



Enlarged view



Make sure to pick the `shelltool` icon above, not one of the other two icons on the screen.[7]

In this case, the `shelltool` icon represents a closed, or iconic, `shelltool`. In fact, each icon represents not a window, but a *tool,* in other words, a window application program that creates windows.

Note: A *shell*, or *command interpreter*, is the piece of software that translates the commands you type so the UNIX operating system can understand and act on them.

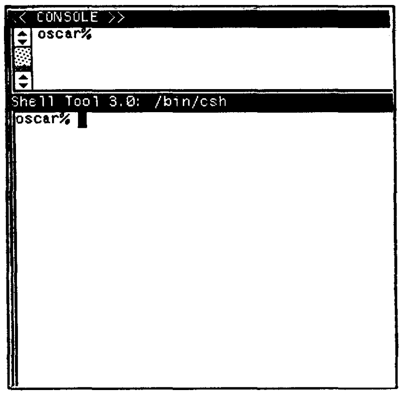
A `shelltool` is one type of tool. `shelltools` run the *shell*, or the *command interpreter*, that you have typed UNIX commands to all along. With `shelltools` that run only within `suntools`, you can have several windows running the UNIX command interpreter on the screen simultaneously.

To *open* the iconic `shelltool` window, move the cursor over the `shelltool` icon and *click*, or press and release, the left mouse button. An open `shelltool` appears in the gray surface of the screen.[8]

[7] You will learn about the `textedit` and `mailtool` icons in the chapter on window-based tools, Chapter 7.

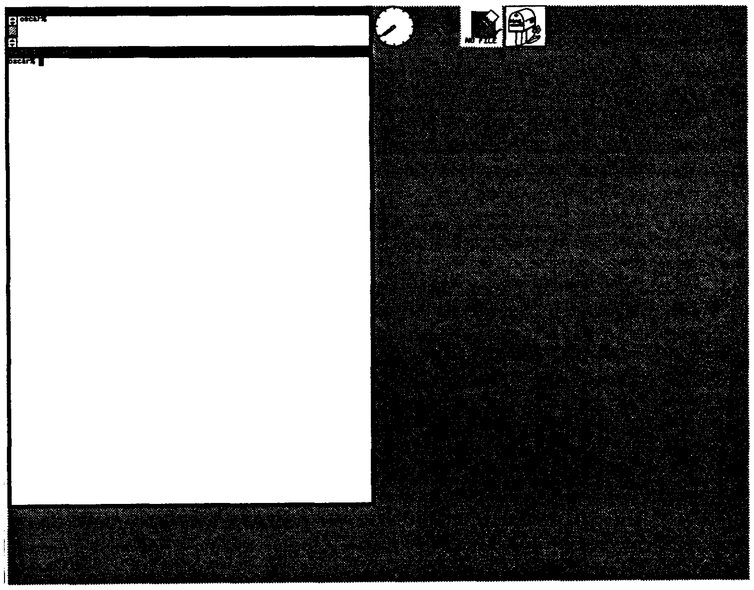
[8] Window system performance may deteriorate when your machine doesn't have enough memory or runs too many programs in the background. When your machine is a client machine of an overloaded file server, performance slows as well. If the window system seems sluggish, contact your system administrator to understand and remedy the difficulty.

Figure 2-6 *Open* `shelltool` *Window*



```
<< CONSOLE >>
oscar%
Shell Tool 3.0: /bin/csh
oscar%
```

Enlarged view



Note: The window *frame* comprises the *namestripe* across the top of the window, and the *borders* on the other three sides of the window.

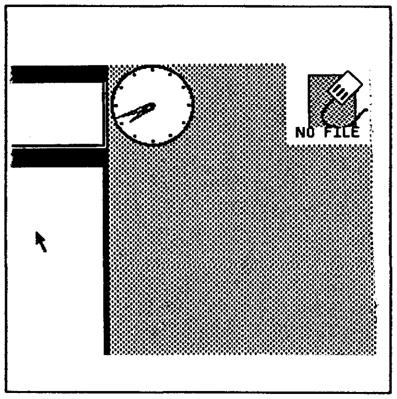
You can use the "left-click" method of opening an iconic window with any of the icons on the screen.

To make use of the window, you must move the cursor within the borders of the window. Locating the cursor on the window *frame* will **not** allow you to type in the window — you must move the cursor **entirely inside** the window.

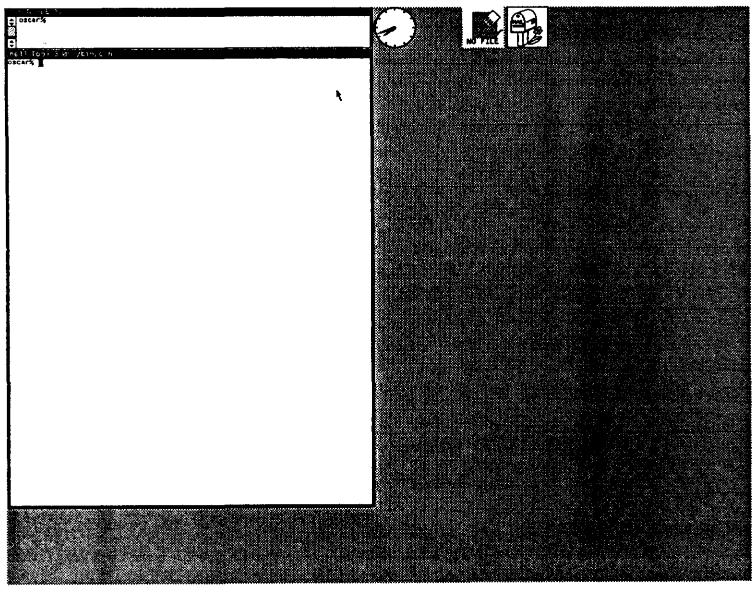
The system confirms your window choice by filling in the window border. Try moving the cursor slowly in and out of the window to see how the window border changes.[9]

[9] If the system does not confirm your window choices by darkening the window border, it may be operating with the "click-to-type" method for choosing windows. See the chapter on focusing attention (Chapter 14 .)

Figure 2-7 *Border Changes When Choosing a Window*



Enlarged view



You can type commands to the command prompt within the window just as you typed commands to the prompt you used before starting `suntools`. For example, try creating a file using `cat` >, looking at it with `more`, and listing your working directory with `ls`. Notice that whenever the cursor wanders outside of the window, your typing no longer appears in the window. You can't do work in that window until you move the cursor into the window again.[10]

2.4. Starting `suntools` Automatically Upon Login

If you want `suntools` to start automatically when you log in on your workstation, insert the following lines at the end of your `.login` file:

```
if ( "`tty`" == "/dev/console" ) then
echo "Do you want windows?"
echo "If not type ^C now"
        /usr/bin/suntools
endif
```

Then, when you log in, the system will display

```
Do you want windows?
If not type ^C now
```

on your screen. If you don't type CTRL-C within a second or two, `suntools` will start.

[10] To learn about "click-to-type," an alternative to the "cursor-in-window" method of choosing a window, see the chapter on focusing attention, Chapter 14.

2.5. Summary

Now that you know how to start `suntools`, fix "stuck" mice, and open an iconic window, you can learn how to modify the open window on the screen.

3

Modifying Windows

3 Modifying Windows

This chapter explains how to modify windows, specifically how to:

- change the location of, or *move*, a window
- change the size of, or *resize*, a window
- *open* and *close* windows

3.1. Modifying Your Window

To modify the window you opened in the last chapter, you can use the *frame* menu. The frame menu allows you to modify individual windows.

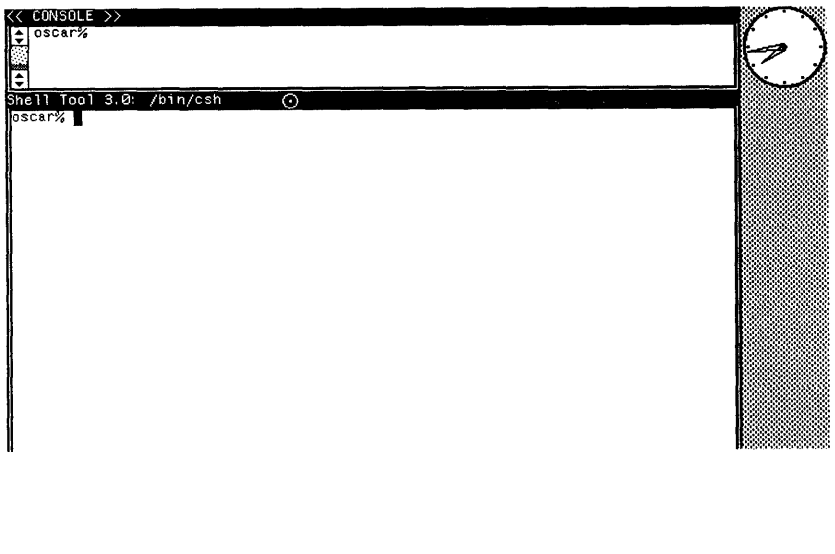
Pop Up the Frame Menu

Note: When you *pop up* a menu, you make it visible on the screen. The frame menu that first appears is an *old-style* menu, unless you change the style, as described in the section on walking menus (Section 8.2), in which case the header `Frame` doesn't appear at the top of the frame menu.

Move the cursor to the namestripe across the top of the window. Then, *pop up* the frame menu by pressing, and holding down, the right mouse button while the cursor is over the window frame (the namestripe or border of the window).

When you move the cursor onto the window frame, a small *target circle cursor* with a dot in the center appears. Pop up the menu by pressing, and holding down, the right mouse button.

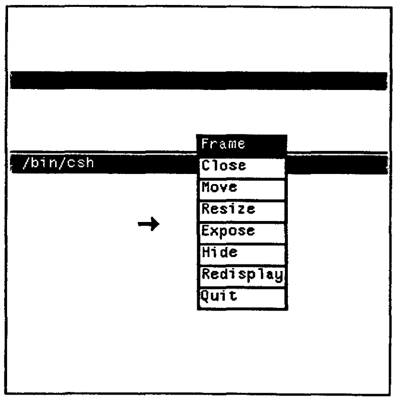
Figure 3-1 *Target Circle for Popping Up a Frame Menu*



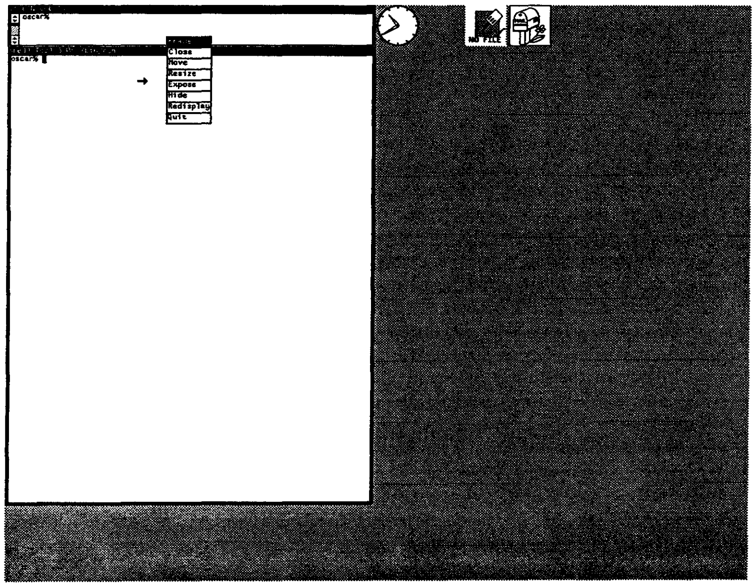
```
oscar%
```

```
oscar%
```

Figure 3-2 *The Frame Menu*



Enlarged view



Note: When you select an item within the menu box, the window system notifies you of your choice by reversing the colors of the rectangular item area you have selected. When you release the mouse button, the window system executes the item you chose. If you don't select anything, nothing happens.

The frame menu has the abbreviation `Frame` as its *header*. Notice that the cursor changes — it points due "east" when you pop up a menu. That is so that you can *choose* a menu item, such as `Move` to change the location of a window.

When you decide not to choose a menu item, move the cursor outside the menu box, and release the right mouse button. The menu box will disappear.

Changing the Location of a Window

How can you move the window from its current screen location to your favorite spot on the vast expanse of unadulterated gray screen?

Here are the steps:

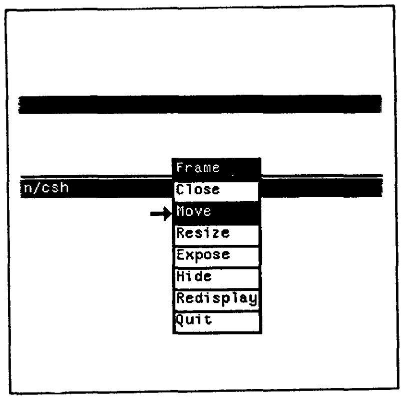
- Pop up a frame menu by pressing the right mouse button while the cursor is in the window namestripe, or targeting a window border
- Keep holding the right mouse button down
- Choose the `Move` item by moving the cursor into the menu item rectangle
- Release the right mouse button (the menu box disappears)
- Read the instruction box that appears on the screen

Note: A *boundary box* is a rectangle that defines the boundary, or borders of a window. So, when changing window locations, the boundary box informs you where the window will settle after you release the mouse button.

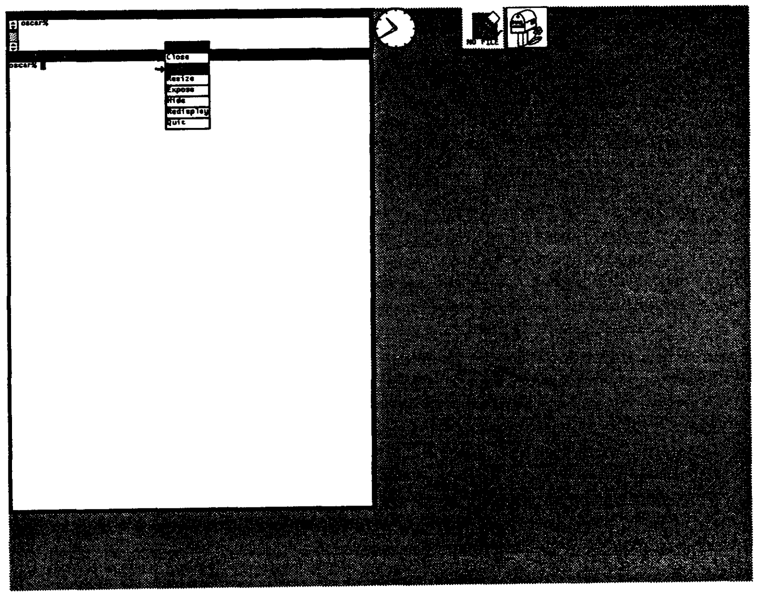
- Drag the window to the desired location by pressing, and holding, the left or middle mouse button and using the mouse to move the window's *boundary box*
- Release the mouse button[11]

Here are some illustrations of these steps:

Figure 3-3 *Choosing the* `Move` *Item in the Frame Menu*

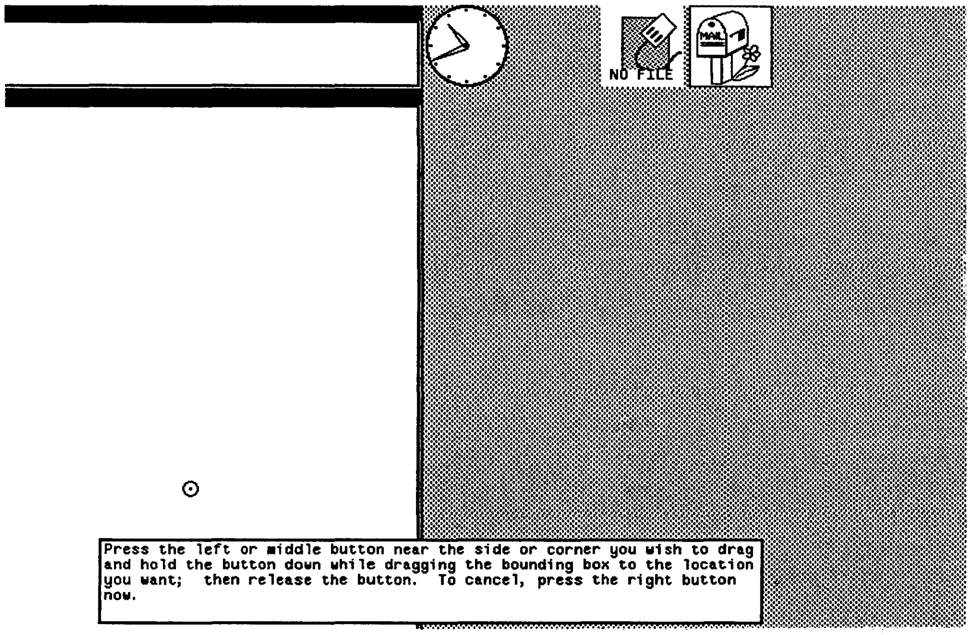


Enlarged view



[11] For a quick method of changing window location without using a menu, see the chapter on accelerators, Chapter 13 . The quick reference appendix provides a quick reference to the function keys and most of the accelerators.

Figure 3-4 *The Instruction Box for Moving Windows*

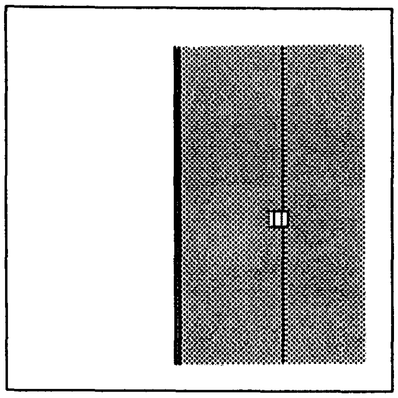


```
Press the left or middle button near the side or corner you wish to drag
and hold the button down while dragging the bounding box to the location
you want;  then release the button.  To cancel, press the right button
now.
```

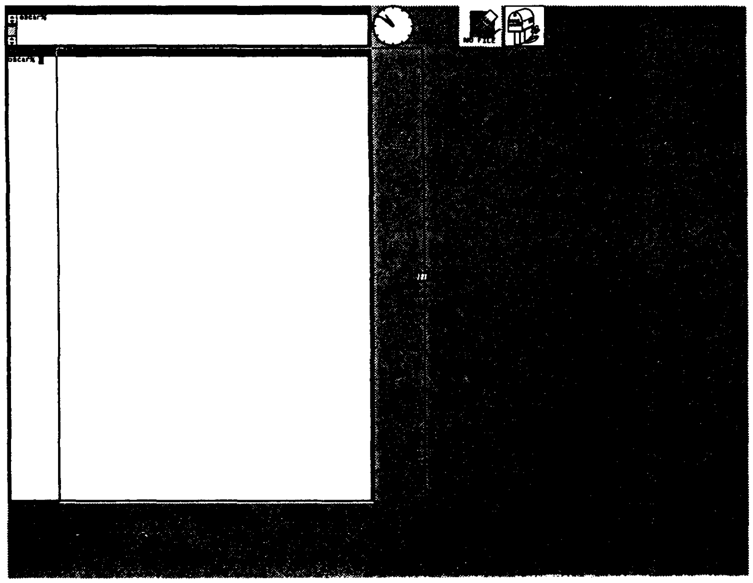
You can *drag* the window boundary box in one of two ways, *constrained* or *unconstrained*, depending on where on the window frame you put the cursor.

When you press the left or middle mouse button while the cursor is on the middle third of the window namestripe or any window edge, you can drag the window side, or edge, along a constrained path. The constrained path is horizontal when you pick the left or right window edge; it is vertical when you pick the namestripe or the bottom edge of the window. You move the mouse so that the *constrained movement cursor* indicates the new window position, then release the mouse button.

Figure 3-5 *Dragging a Window Boundary Box: Constrained*

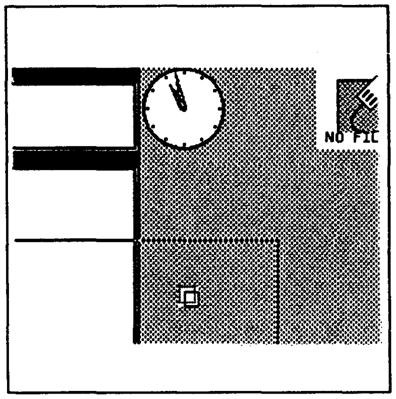


Enlarged view

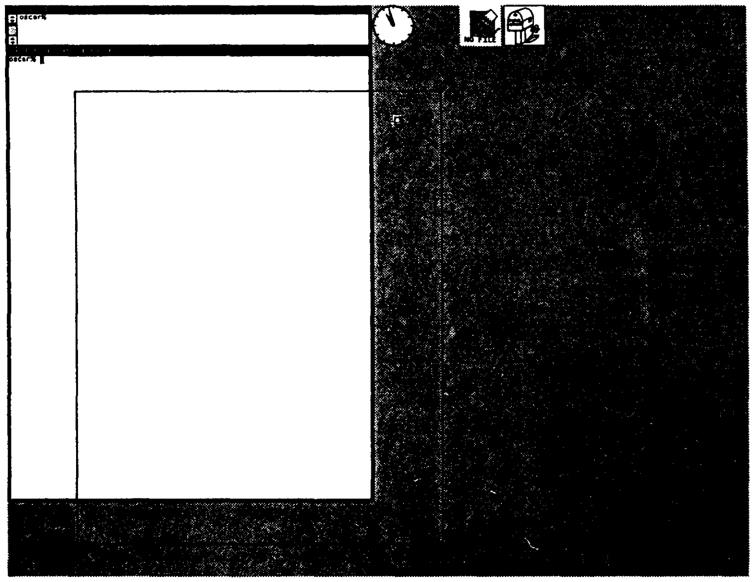


When instead you press the left or middle mouse button while the cursor is within one-third edge length of a window corner, you can drag the window corner along an unconstrained path. The unconstrained path is horizontal, vertical, or diagonal, depending on how you move the ***unconstrained movement cursor***. Release the mouse button to fix the window in the new position that you desire.

Figure 3-6 *Dragging a Window Boundary Box: Unconstrained*



Enlarged view



When you can see the instruction box and you want to cancel the moving operation, simply click the right mouse button, as indicated in the instructions, and the instruction box will disappear without changing the location of the window.

Change the location of this window a few times, so you can familiarize yourself with the process.

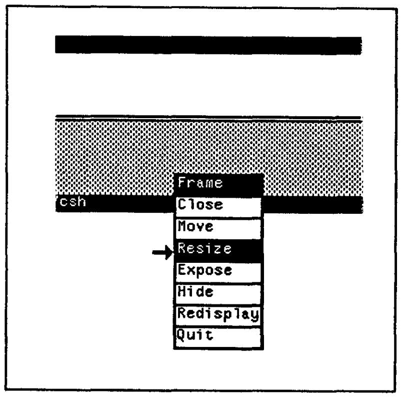
Resizing, or Changing the Size of a Window

Resizing, or changing the size of a window, is similar to moving a window. Follow these steps:

- Pop up a frame menu
- Choose the `Resize` item in the menu
- Release the right mouse button
- Read the instruction box that appears on the screen
- Specify the desired window size by pressing the left or middle mouse button and using the mouse to move the window's *boundary box*
- Release the mouse button

Here are some illustrations:

Figure 3-7 *Choosing the* `Resize` *Item in the Frame Menu*



Enlarged view

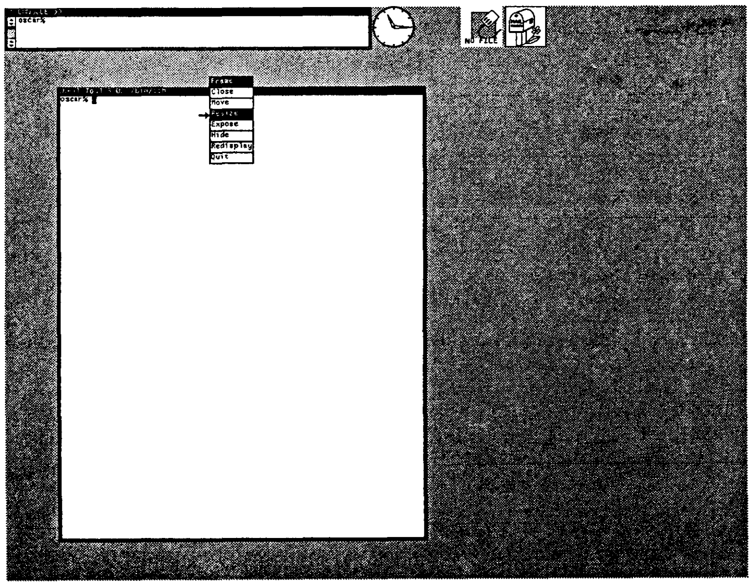
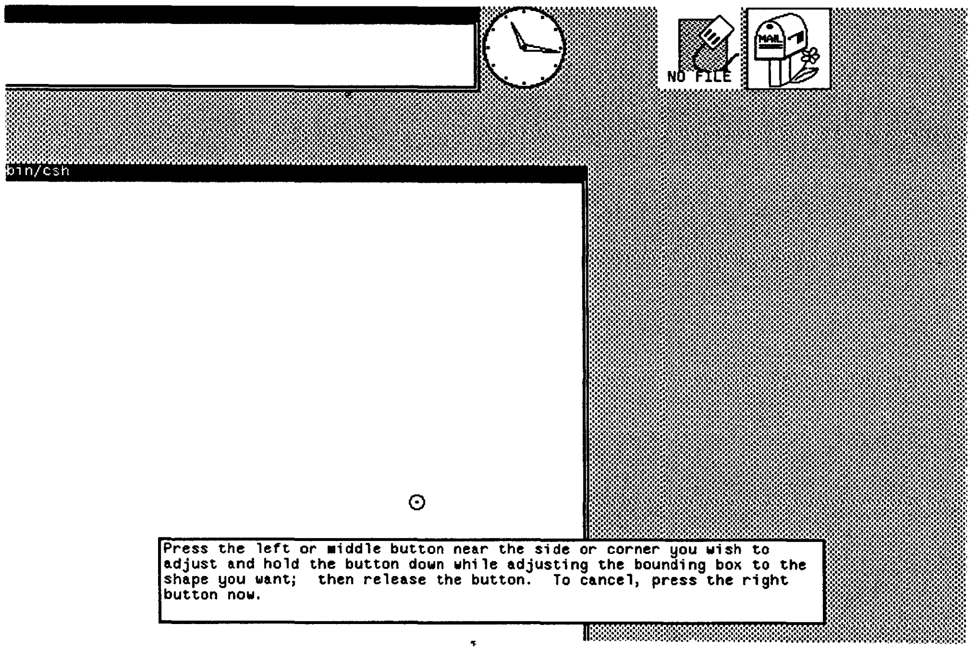


Figure 3-8 *The Instruction Box for Resizing Windows*

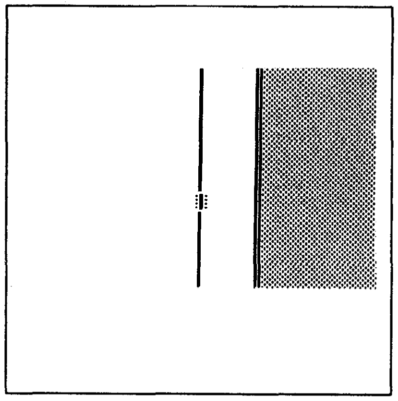


Press the left or middle button near the side or corner you wish to adjust and hold the button down while adjusting the bounding box to the shape you want; then release the button. To cancel, press the right button now.

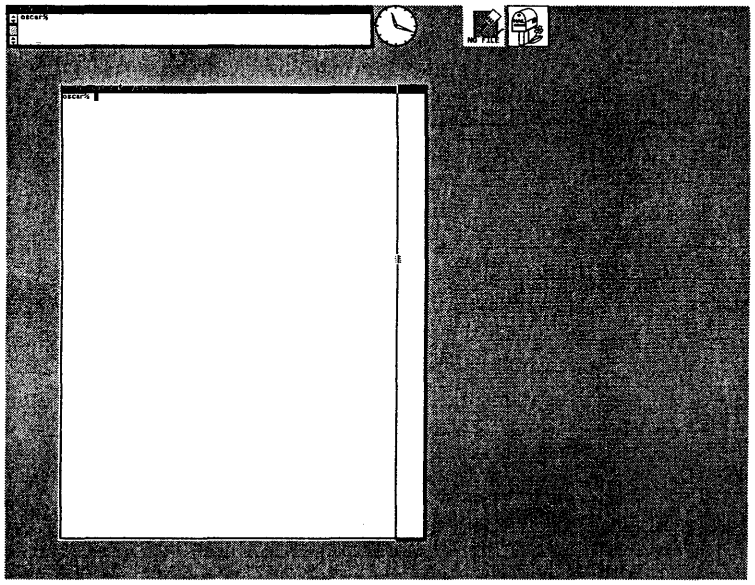
You can *adjust* the window boundary box in one of two ways, *constrained* or *unconstrained*, depending on where on the window frame you put the cursor.

When you press the left or middle mouse button while the cursor is on the middle third of the window namestripe or any window edge, you can adjust the window side, or edge, along a constrained path. The constrained path is horizontal when you pick the left or right window edge; it is vertical when you pick the namestripe or the bottom edge of the window. You move the mouse so that the *constrained resizing cursor* indicates the new window position, then release the mouse button.

Figure 3-9 *Adjusting a Window Boundary Box: Constrained*

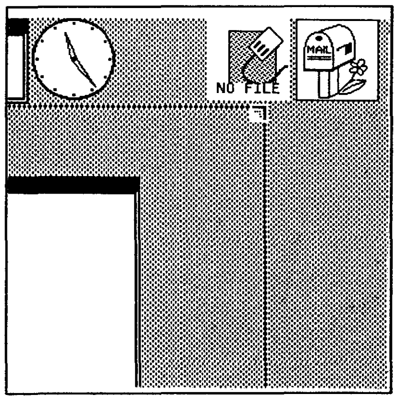


Enlarged view

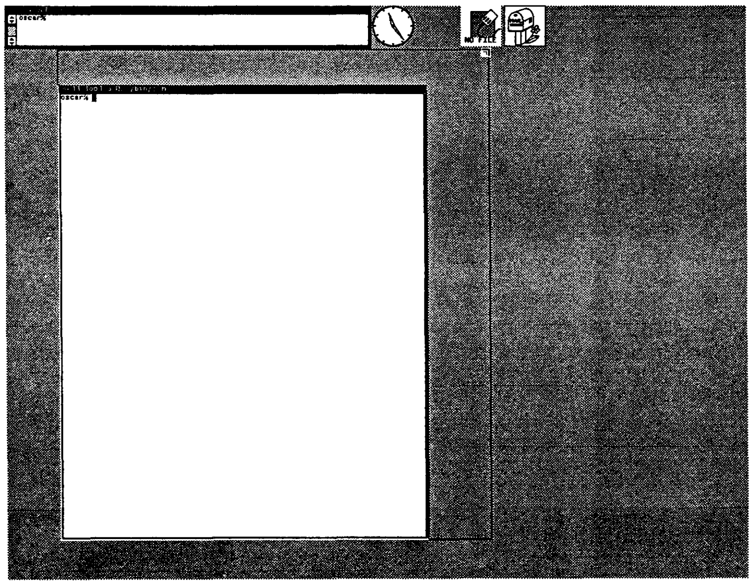


When instead you press the left or middle mouse button while the cursor is within one-third edge length of a window corner, you can adjust the window corner along an unconstrained path. The unconstrained path is horizontal, vertical, or diagonal, depending on how you move the *unconstrained resizing cursor.* Release the mouse button to fix the window in the new position that you desire.

Figure 3-10 *Adjusting a Window Boundary Box: Unconstrained*



Enlarged view



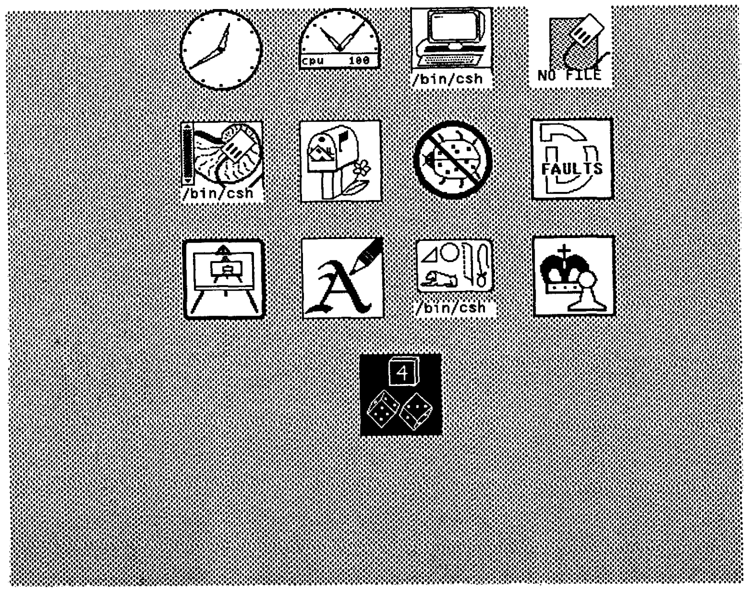
The instruction box notes that you can cancel the resizing operation by clicking the right mouse button, rather than continuing to size the window boundary box.[12]

Closing and Opening a Window

Sometimes it is useful to "close" a window, or transform it into an *icon*. You might want to close the window while you aren't using it, so that the window doesn't take up as much space on the screen.

An icon is a little drawing that represents a closed window. Each type of window has its own icon. (You'll read more about types of windows in Chapter 7 on window-based tools.)

Figure 3-11 *Some Common Icons*



You can type commands to open windows, but not to closed windows.[13] However, an iconic window will continue to execute commands that you typed before you closed the window.

To close a window, follow these steps:

- Pop up a frame menu
- Choose the `Close` item in the menu
- Release the right mouse button[14]

[12] For a quick way to resize windows without using a menu, see the chapter on accelerators, Chapter 13. The quick reference appendix provides a quick reference to the function keys and most of the accelerators.

[13] `perfmeter` is one exception; see its Man Page, online or in the *Commands Reference Manual*.

[14] For a quick way to close windows without using a menu, see the chapter on accelerators, Chapter 13. The quick reference appendix provides a quick reference to the function keys and most of the accelerators.

Figure 3-12 *Open* `shelltool` *Window*

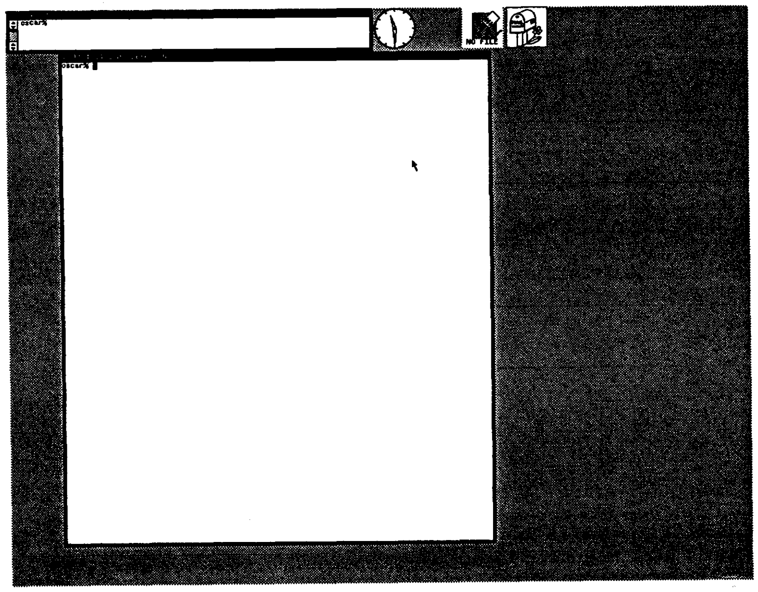
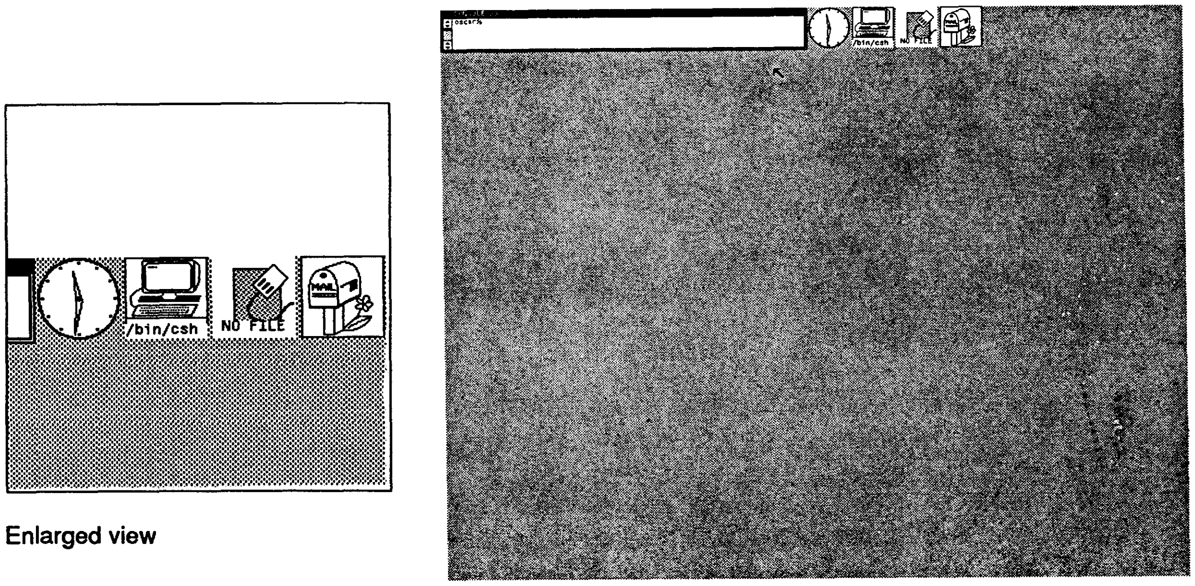


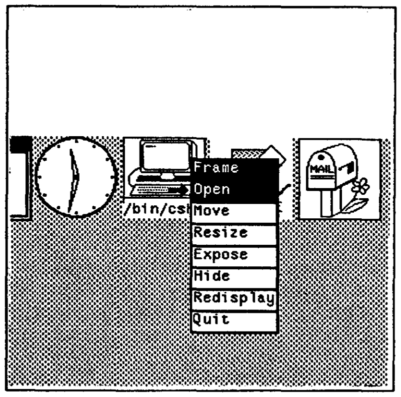
Figure 3-13 *Closed, or Iconic,* `shelltool` *Window*



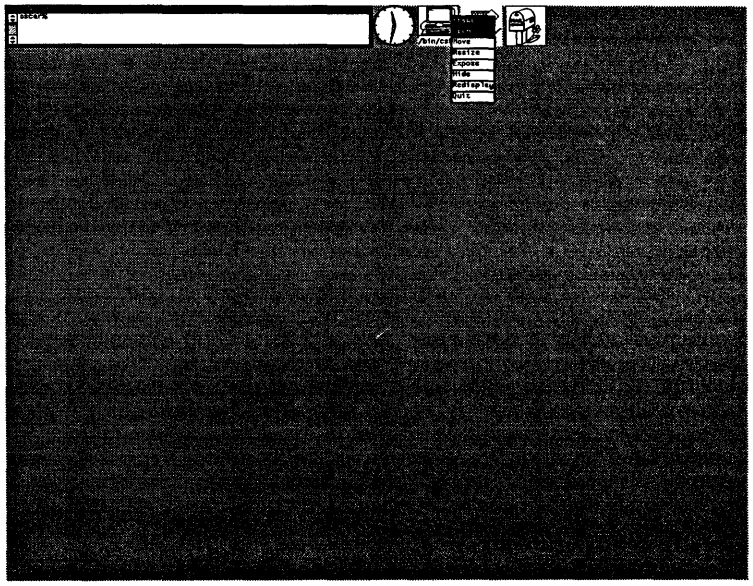
To open a window that is in its iconic state, follow these steps:

- Pop up a frame menu by pressing the right mouse button while the cursor is anywhere in the iconic rectangle
- Choose the `Open` item in the menu
- Release the right mouse button

Figure 3-14 *Choosing the* `Open` *Item in the Frame Menu*



Enlarged view



When you first started to use the window system, you opened a window by clicking the left button once on the appropriate icon. That is a fast way to open a window, known as an *accelerator*.[15]

Try opening and closing the window a couple of times to make sure you've got the idea.

You've learned how to move, resize, open, and close a window. Next, you can learn how to create a window "from scratch."

[15] See the chapter about accelerators, Chapter 13 . The quick reference appendix provides a quick reference to the function keys and most of the accelerators.

4

Creating Application Windows

4

Creating Application Windows

4.1. Creating Application Windows

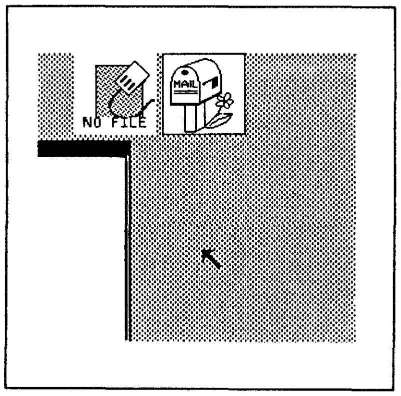
When you want more than one `shelltool`, or when you don't see a `shelltool` icon on the screen, you can create a `shelltool` ''from scratch.''

Pop Up the Root Menu

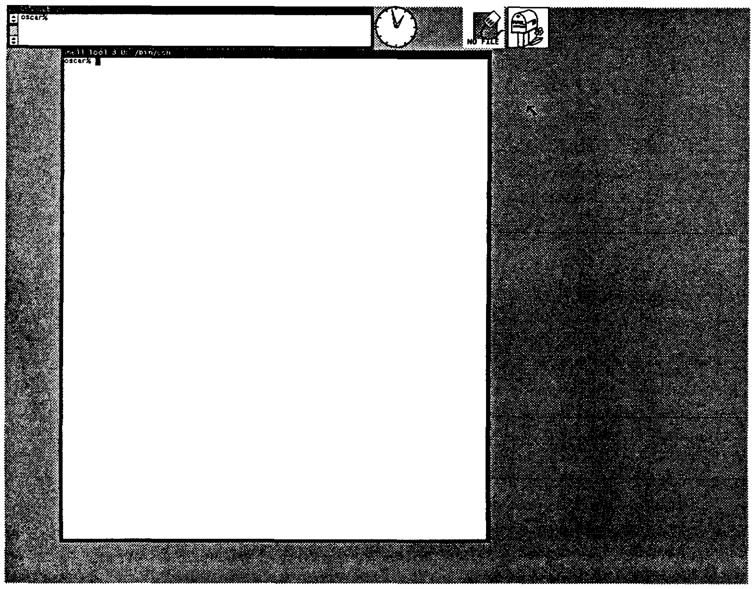
To create a new `shelltool`, use another menu, called the *root menu.* You can use the root menu to create application windows, exit `suntools`, and a variety of other functions.

For now, however, you want to create a `shelltool`. To pop up the root menu, move the cursor onto a gray area of the screen, approximately where you want the menu to appear.

Figure 4-1 *Press the Right Mouse Button When the Cursor Is On Gray*

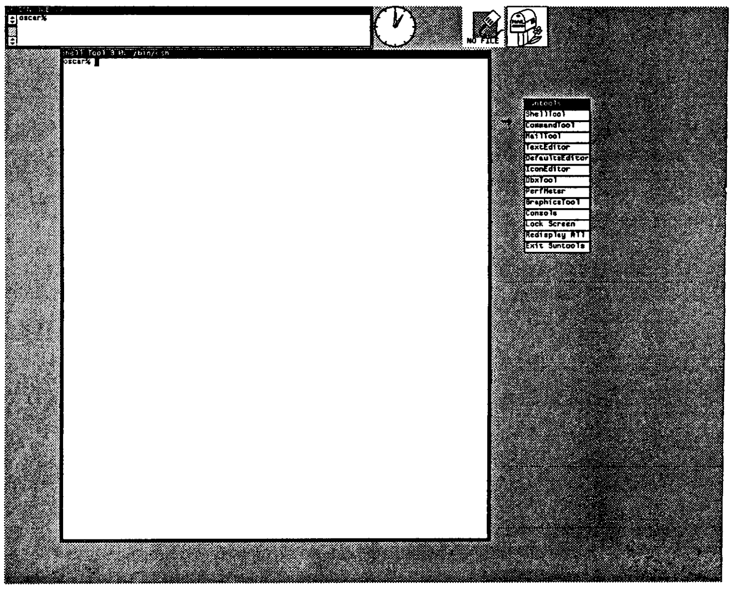


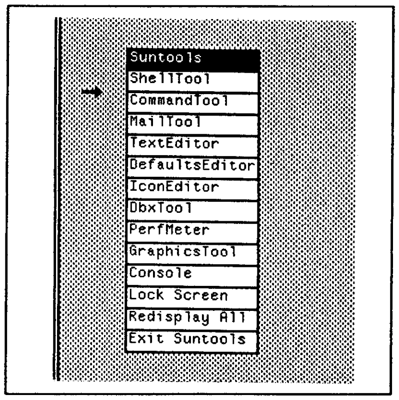
Enlarged view



Then, press the mouse button on the right side of the mouse and hold it down. A menu that looks like this appears:

Figure 4-2 *The Root Menu*





Enlarged view

This is the root menu, with the word `Suntools` as its *header*. Just as with the frame menu, the cursor changes — it points due east when you pop up a menu. That is so that you can *choose* a menu item, such as `ShellTool`, to create a window.

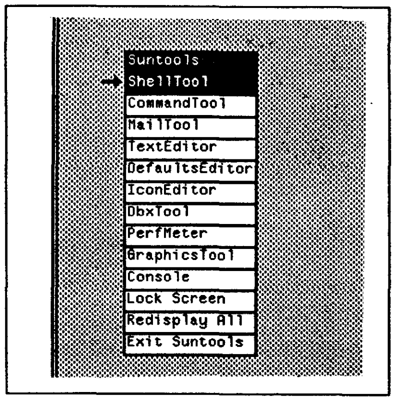
When you decide you don't want to choose any of the menu items, just move the cursor away from the menu box and release the right mouse button. The menu will disappear.

4.2. Creating a Window

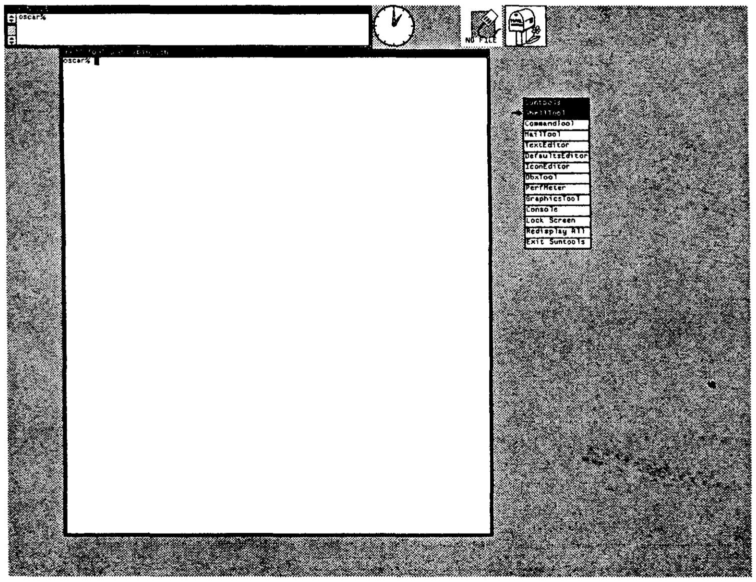
To create a `shelltool`, you must choose the `ShellTool` entry, the first item in the root menu. Here are the steps:

- Pop up the menu (press the right mouse button)
- Keep holding the mouse button down
- Move the cursor into the `ShellTool` rectangle within the menu box
- Release the mouse button

Figure 4-3 *Choosing* `ShellTool` *Item on Root Menu*



Enlarged view

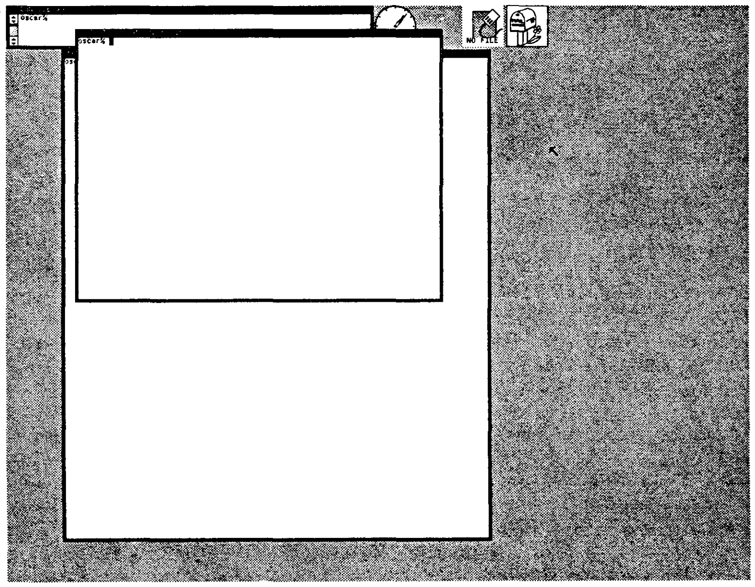


A new window appears, usually in the upper center of your screen. At first, the window is empty, with only a *namestripe* describing the window, for instance:

```
Shell Tool 3.0: /bin/csh
```

Then, the command prompt appears.

Figure 4-4 *Creating a Window*



You have created your first window using the window system.

5

Manipulating Windows

5

Manipulating Windows

The chapter explains how to:

- expose and hide windows
- redisplay a window, or all of the windows
- quit a window, or exit the entire window system

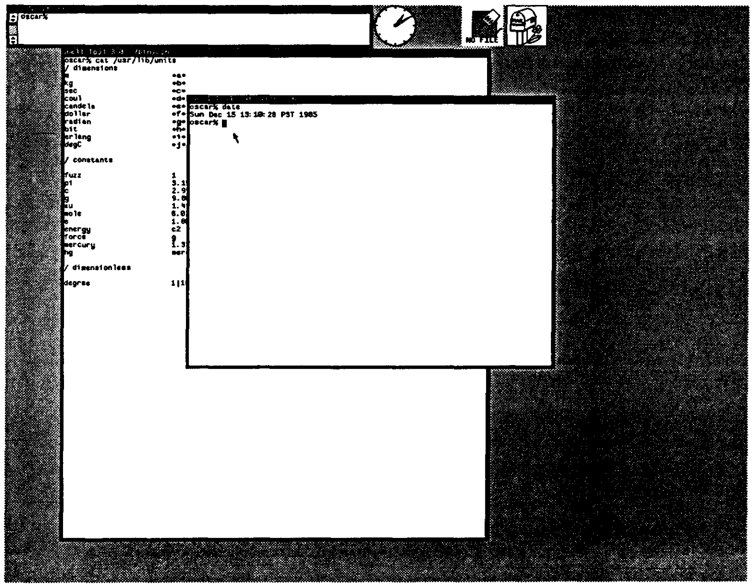
5.1. Playing With Multiple Windows

A big advantage of the window system is that you can create more than one window and run different commands and programs simultaneously, each program in a window you have created.

Note: When you move the cursor into a window to start typing, the window border shading changes, confirming your choice of window.

Move the `shelltool` window you created at the end of the last chapter so that you can type to the command prompts on both windows. Then, move the cursor into one `shelltool` window, and `cat` a long file, like `/usr/lib/units`. Next, move the cursor into the other window and type `date` to find out the current date and time.

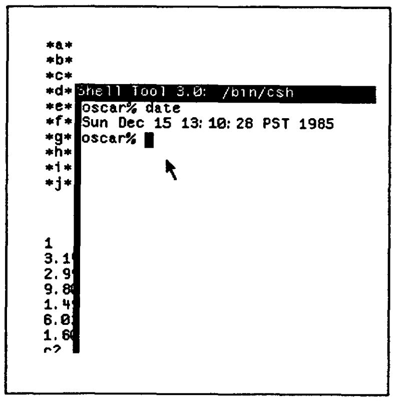
Figure 5-1 *Playing With Multiple Windows*



```
oscar%
```

```
oscar% cat /usr/lib/units
/ dimensions
m                       *a*
kg                      *b*
sec                     *c*
coul                    *d*
candela                 *e*
dollar                  *f*
radian                  *g*
bit                     *h*
erlang                  *i*
degC                    *j*

/ constants

fuzz                    1
pi                      3.1
c                       2.9
g                       9.8
au                      1.4
mole                    6.0
e                       1.6
energy                  c2
force                   g
mercury                 1.3
hg                      mer

/ dimensionless

degree                  1|1
```

```
oscar% date
Sun Dec 15 13:10:28 PST 1985
oscar% █
```



```
*a*
*b*
*c*
*d* Shell Tool 3.0:  /bin/csh
*e* oscar% date
*f* Sun Dec 15 13:10:28 PST 1985
*g* oscar% █
*h*
*i*
*j*


1
3.1
2.9
9.8
1.4
6.0
1.6
c?
```

Enlarged view

One window continues displaying the file, while you type to the other window. You can work more efficiently by running multiple programs simultaneously in various windows.

Exposing Windows

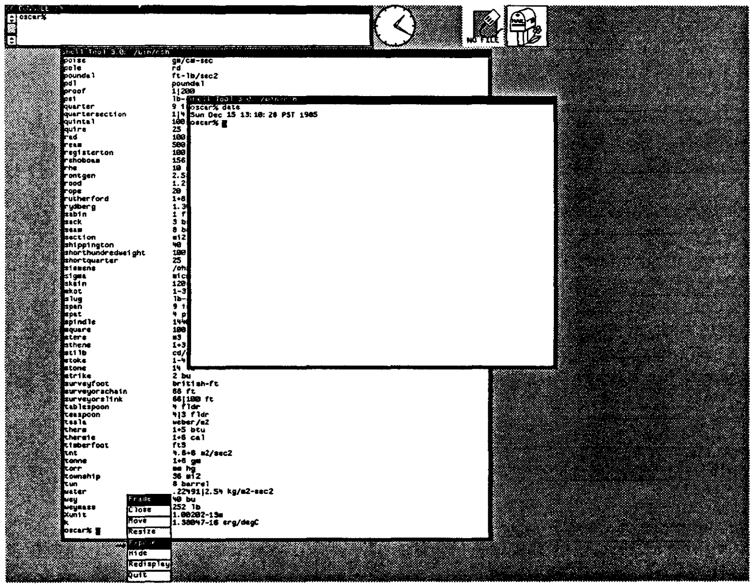
When you create new windows, sometimes they may overlap. One way to make visible a window that is "hidden" behind another window is to *move* that window out from behind the other window. Or, you can *expose* the window that is hidden, using the `Expose` item on the frame menu. Exposing a window is like shuffling a piece of paper from the middle or bottom of a pile of papers on your desk to the top of the pile.

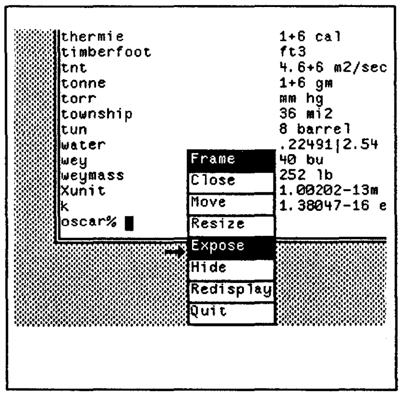
When you want to expose a window, follow these steps:

- Move the cursor onto the border or namestripe of the window to get the frame menu
- Choose the `Expose` item in the menu box
- Release the right mouse button you used to pop up the menu

Each time you create a window, it appears on top of the pile. To try exposing a window, move the window that is on top of the pile — in other words, the window you just created — so that it covers, or hides, another window. Try exposing the window that is hidden behind the one you just created.

Figure 5-2 *Exposing a Window: Choosing the* `Expose` *Menu Item*



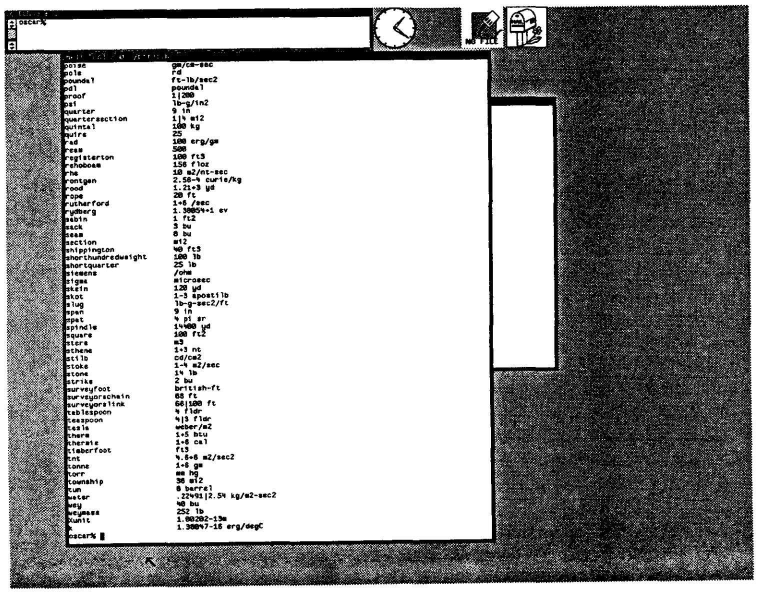


```
thermie                  1+6 cal
timberfoot               ft3
tnt                      4.6+6 m2/sec
tonne                    1+6 gm
torr                     mm hg
township                 36 mi2
tun                      8 barrel
water                    .22491|2.54
wey                      40 bu
weymass                  252 lb
Xunit                    1.00202-13m
k                        1.38047-16 e
oscar% █
```

Enlarged view

The window you expose shuffles to the top of the pile of windows on the screen.

Figure 5-3 *Exposing a Window: After It's Exposed*



```
oscar%
```

```
poise                   gm/cm-sec
pole                    rd
poundal                 ft-lb/sec2
pdl                     poundal
proof                   1|200
psi                     lb-g/in2
quarter                 9 in
quartersection          1|4 mi2
quintal                 100 kg
quire                   25
rad                     100 erg/gm
ream                    500
registerton             100 ft3
rehoboam                156 floz
rhe                     10 m2/nt-sec
rontgen                 2.58-4 curie/kg
rood                    1.21+3 yd
rope                    20 ft
rutherford              1+6 /sec
rydberg                 1.36054+1 ev
sabin                   1 ft2
sack                    3 bu
seam                    8 bu
section                 mi2
shippington             40 ft3
shorthundredweight      100 lb
shortquarter            25 lb
siemens                 /ohm
sigma                   microsec
skein                   120 yd
skot                    1-3 apostilb
slug                    lb-g-sec2/ft
span                    9 in
spat                    4 pi sr
spindle                 14400 yd
square                  100 ft2
stere                   m3
sthene                  1+3 nt
stilb                   cd/cm2
stoke                   1-4 m2/sec
stone                   14 lb
strike                  2 bu
surveyfoot              british-ft
surveyorschain          66 ft
surveyorslink           66|100 ft
tablespoon              4 fldr
teaspoon                4|3 fldr
tesla                   weber/m2
therm                   1+5 btu
thermie                 1+6 cal
timberfoot              ft3
tnt                     4.6+6 m2/sec2
tonne                   1+6 gm
torr                    mm hg
township                36 mi2
tun                     8 barrel
water                   .22491|2.54 kg/m2-sec2
wey                     40 bu
weymass                 252 lb
Xunit                   1.00202-13m
k                       1.38047-16 erg/degC
oscar% █
```

Hiding Windows

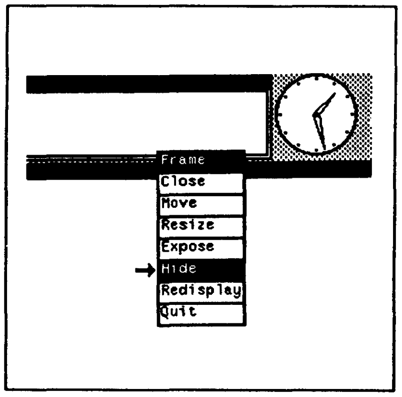
At other times, you may want to *hide* a window behind some others because you don't need it for a while, just as you might shuffle a sheet of paper to the bottom of the pile of papers on your desk.

To try hiding a window, move the window that is on top of the pile, so that it covers, or hides, another window.

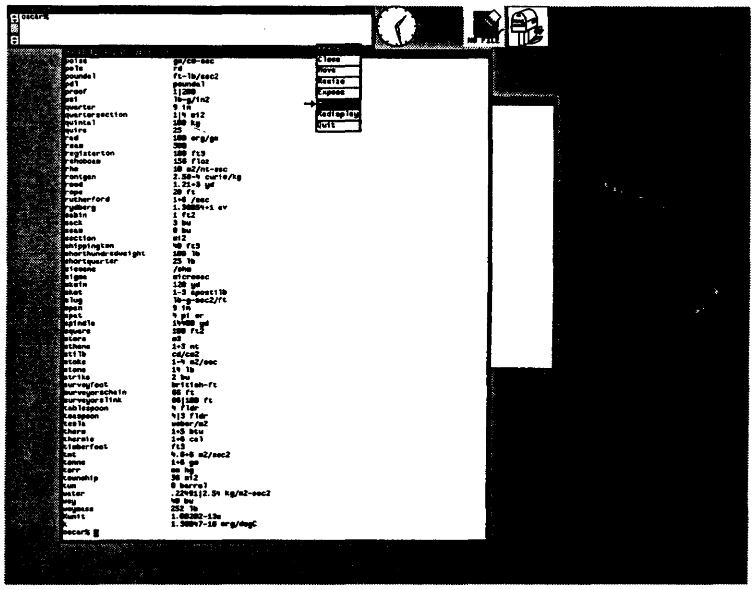
Now, hide the front window behind the window it obscures by following these steps:

- Move the cursor onto the border or namestripe of the window to get the frame menu
- Choose the `Hide` item in the menu box
- Release the right mouse button you used to pop up the menu

Figure 5-4 *Hiding a Window: Choosing the* `Hide` *Menu Item*

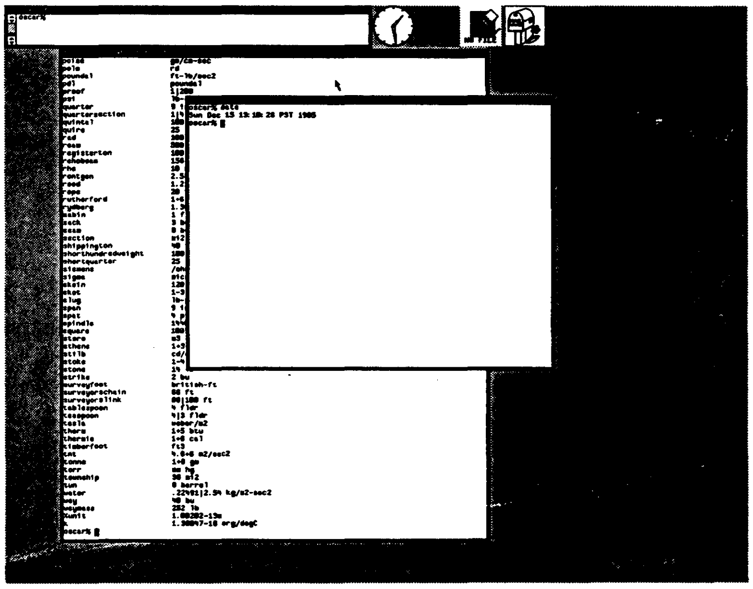


Enlarged view



The window you hide shuffles to the bottom of the pile of windows on the screen.[16]

Figure 5-5 *Hiding a Window: After It's Hidden*



[16] For a quick way to expose and hide windows, see the chapter on accelerators, Chapter 13 . The quick reference appendix provides a quick reference the function keys and most of the accelerators.

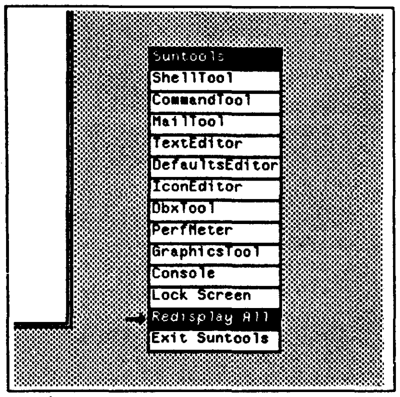
Redisplaying a Window

Occasionally, the contents of a window may become garbled. You can *redisplay*, or draw the window again, by choosing the `Redisplay` item on its frame menu.

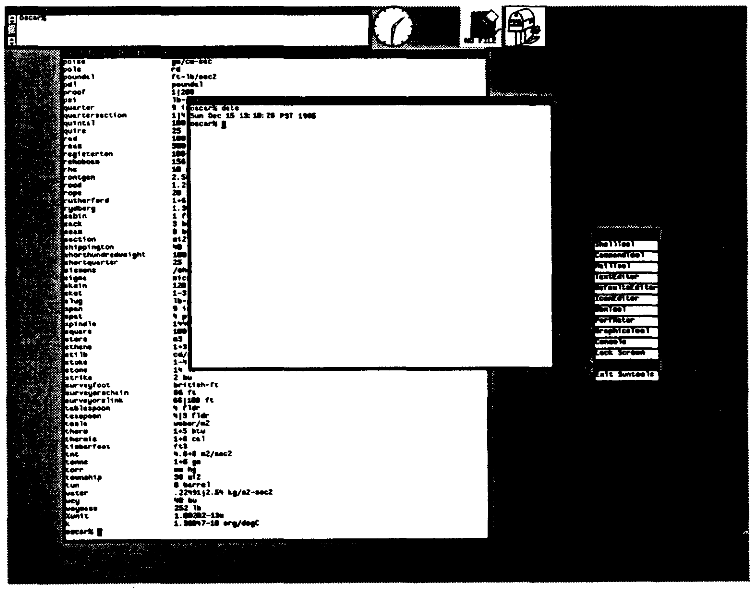
Redisplaying All of the Windows

If most or all of the windows on the screen become garbled, redisplay the entire screen by choosing the `Redisplay All` item in the root menu.

Figure 5-6 *Redisplaying All of the Windows on the Screen*



Enlarged view



Note: Put only one console window on your screen at a time to prevent confusion.

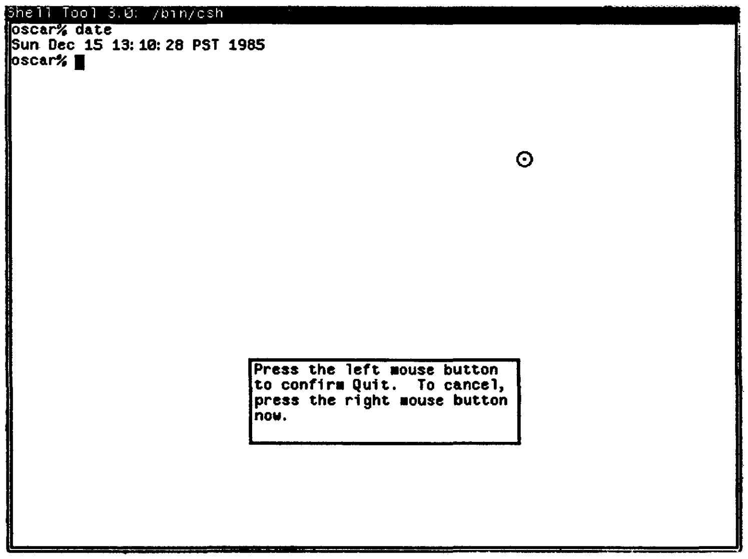
Always keep a console window on your screen, so that system messages don't flash onto the entire screen, or alternate windows, disturbing your work. When you don't have a console window on the screen, you can create one by picking the `Console` entry on the root menu.

Quitting a Window

When you want to get rid of, or *quit*, a specific application, choose the `Quit` item on its frame menu. The window system displays an instruction box and asks you to click the left mouse button to confirm the quit, or the right mouse button to abort the quit. When you confirm, the window disappears. When you abort, the window remains.[17]

[17] For now, there is no accelerator, or quick way, to quit windows.

Figure 5-7 *Quitting a Window: Instruction Box*



```
oscar% date
Sun Dec 15 13:10:28 PST 1985
oscar% █
```

```
Press the left mouse button
to confirm Quit.  To cancel,
press the right mouse button
now.
```

When possible, try to exit from any programs you have running in the window before quitting the window.

Exiting the Window System

Sometimes, you may want to exit `suntools`, getting rid of all the windows and the gray screen background at once. Rather than quitting each individual tool, you can choose the `Exit Suntools` item on the root menu to exit `suntools` entirely. Once again, the window system will print the confirmation instruction box. When you want to confirm the exit, click the left mouse button. When you want to abort the exit, preserving all of your window tools for further use, click the right mouse button.

Before exiting the window system, make sure that any long-running programs have finished execution, so that you don't destroy the partial result of the program.

In this chapter, you learned how to run programs simultaneously in more than one window, and how to hide, expose, redisplay, and quit a window, or the window system. You've covered the basics of window-based tool setup, modification, creation, and manipulation.

6

Storing Window Attributes

6

Storing Window Attributes

After you experiment with the window system for a while, you may wonder how to store the attributes of the window-based tools that you have created, including the window locations and sizes, for reuse. When you store these window attributes, then each time you start `suntools` you can continue with the window set up the way they were when you stored them.

6.1. `toolplaces`

`toolplaces` is a command that displays the attributes of the windows that you have on the screen. Type `toolplaces` to see what it does.

Figure 6-1 *Running* `toolplaces`

Note: The line starting with `-WL` "wrapped around," continuing from the end of the previous line.

```
venus% toolplaces
cmdtool     -Wp    0   0 -Ws 570  67 -WP    0   0 -Wl "<< CONSOLE >>"
 -WL console -C
clock       -Wp  568   0 -Ws 192  31 -WP  572   0 -Wi
textedit    -Wp  283 165 -Ws 584 576 -WP  708   0 -Wi
mailtool    -Wp  568  66 -Ws 584 555 -WP  776   0 -Wi
shelltool   -Wp  106  67 -Ws 638 790 -WP  640   0
shelltool   -Wp  126 164 -Ws 570 427 -WP   64   0
venus%
```

Note: The filename `.suntools` begins with a dot, or period character, so it won't show up in a regular file listing with `ls`, unless you specify explicitly the filename. See the example listing below.

You can redirect the output of `toolplaces` into a file. By redirecting the output from `toolplaces` into a file called `.suntools` in your home directory, you can store the current attributes of the windows on the screen. When starting, `suntools` looks at the `.suntools` file in your home directory, if you have one, so it can reproduce the window attributes you store there.

Note: Remember that the tilde character (~) is an abbreviation for your home directory.

To store your current window attributes, create all of the windows you want, move them, resize them, and open or close them, then type `toolplaces > ~/.suntools` to a command prompt in one of the `shelltools`. Make sure to put a dot (`.`) at the beginning of the filename `.suntools`.[18]

Figure 6-2 *Storing Window Attributes with* `toolplaces`

```
venus% toolplaces > ~/.suntools
venus% ls ~/.suntools
/usr/medici/.suntools
venus%
```

Exit the window system by choosing `Exit Suntools` on the root menu. Then, restart the window system by typing `suntools` to your command prompt. `suntools` looks at the `.suntools` file to decide how to set up the windows on your screen.[19]

Give the windows time to appear. The window system should display all of the windows, in the appropriate locations and sizes, that you had on the screen when you stored them with `toolplaces`.

6.2. The `.suntools` File

After creating a `.suntools` file with `toolplaces`, you can look at it. This section describes the format of the `.suntools` file.

Here is what the `.suntools` file would look like if you stored your window attributes with `toolplaces` just after starting `suntools`.

Figure 6-3 *Default* `.suntools` *File*

```
venus%  cat ~/.suntools
cmdtool     -Wp   0    0 -WP    0    0 -Wh  4 -Ww 80
 -Wl "<< CONSOLE >>" -WL "console" -C (Continuation of the first line.)
clock       -Wp 568    0 -WP  572    0 -Wi -Wh 1
shelltool   -Wp   0   66 -WP  640    0 -Wi -Wh 58 -Ww 80
textedit    -Wp 283  165 -WP  708    0 -Wi
mailtool    -Wp 568   66 -WP  776    0 -Wi
venus%
```

Each line of the `.suntools` file can specify:

name	the tool name
`–Wp` *x-position y-position*	location of the window
`–Ws` *pixel-width pixel-height*	size of the window

[18] For more information on `toolplaces`, see the Man Page online, or in the *Commands Reference Manual*.

[19] When you don't have a `.suntools` file in your home directory, `suntools` finds a default `.suntools` file in one of the system directories.

–WP *x-icon-position y-icon-position*	location of the icon
–Wi	iconic (closed if –Wi is present)
–Wl *namestripe-label*	namestripe label
–WL *icon-label*	icon label
–WI *filename*	alternate icon image
–Wt *filename*	alternate font
-Wb/f/g	color control
program	program (only for shelltool and cmdtool; after other specifications)
options	program options (only for shelltool and cmdtool; after other specifications)

For more information on the .suntools file format, see the suntools Man Page, online or in the *Commands Reference Manual.*

Note: You can type to the console window just like any shelltool window, although system messages may occasionally flash onto the window.

When you first started suntools, several iconic windows and a console window appeared, even though you didn't have a .suntools file. When suntools doesn't find a .suntools file in your home directory, it looks for the default .suntools file, /usr/lib/.suntools. Look at the default .suntools file figure above.

You have learned how to store your window attributes for reuse.

7

Window-Based Tools

7 Window-Based Tools

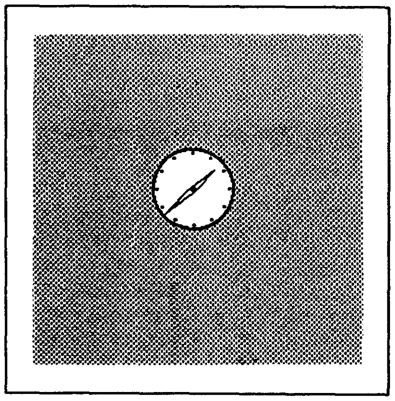
This chapter describes the wide variety of window-based tools supported by the window system. All of the icons that appear at the top of your screen when you you start `suntools` are closed window-based tools of various types. This chapter lists the window-based tools, and describes `clock`, `perfmeter`, and `shelltool`. Later chapters, and other documents, describe the rest of the tools.

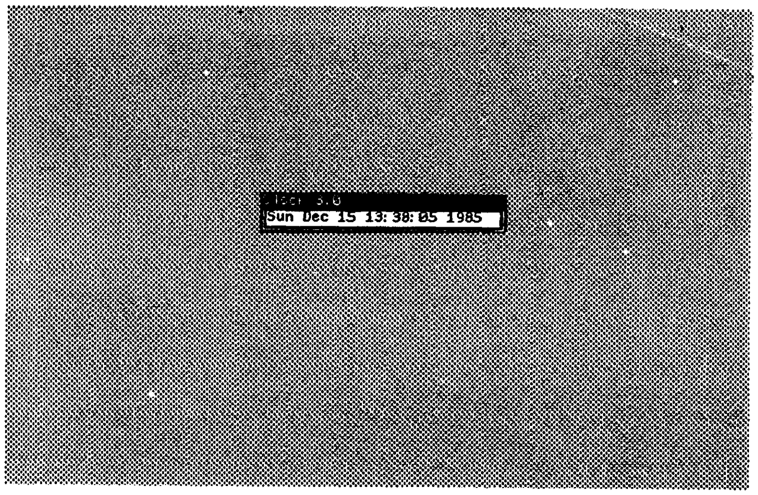
7.1. List of Window-Based Tools

Here is a list of the window-based tools presently available on the Sun system (illustrations of the iconic tool are in the margin; the open tool appears within the body of the text):

`suntools`
: The program that permits all of the window-based tools to run on the workstation screen.

`clock`
: A clock on your screen that displays the current date and time. See Section 7.2 .

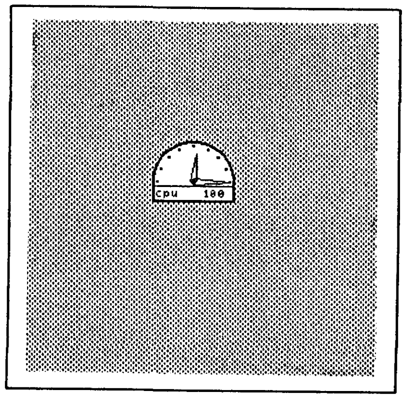

`clock` icon



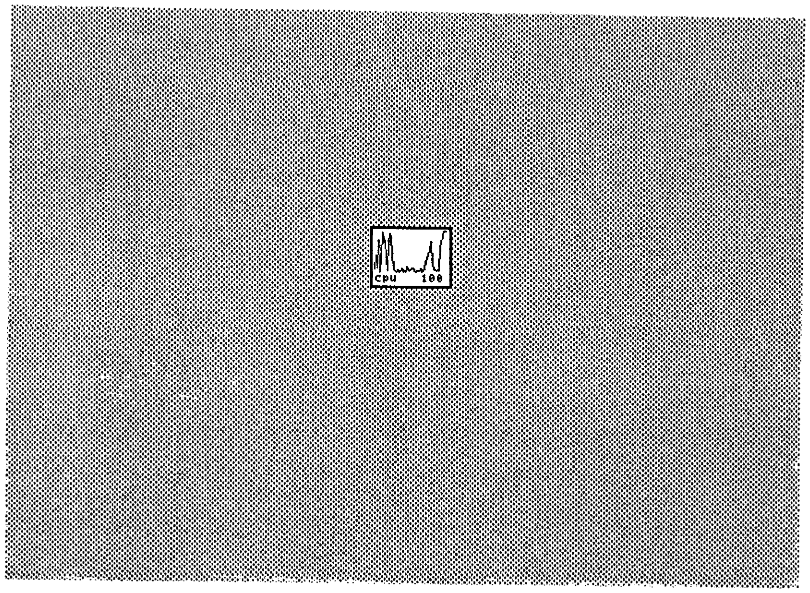
```
clock 3.0
Sun Dec 15 13:38:05 1985
```

`perfmeter`

A "meter" on your screen that displays various system performance statistics. See Section 7.3 .

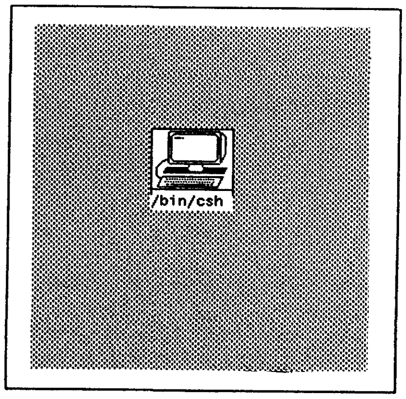


`perfmeter` icon

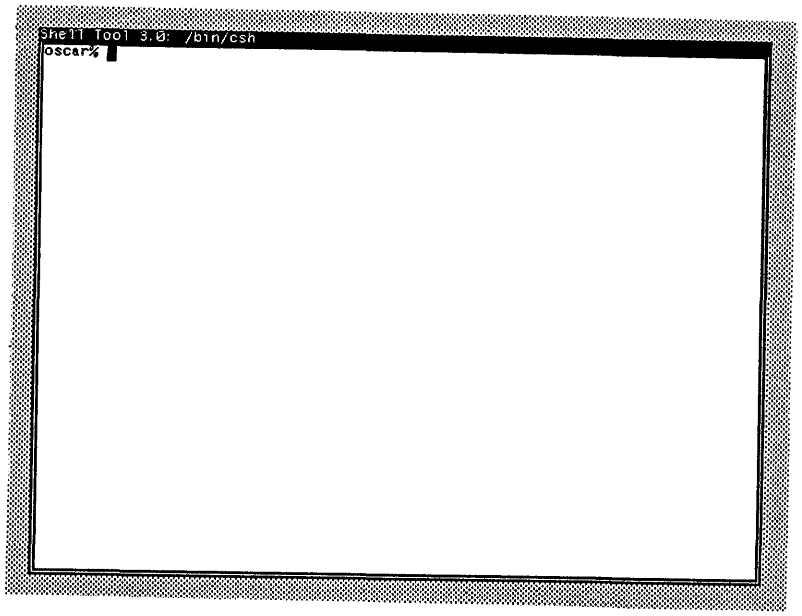


`shelltool`

You've used `shelltool`, a UNIX shell in a window, already. See Section 7.4 .



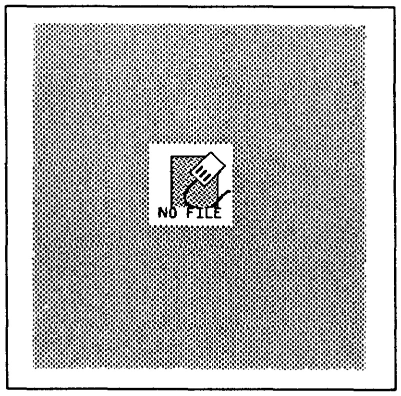
`shelltool` icon



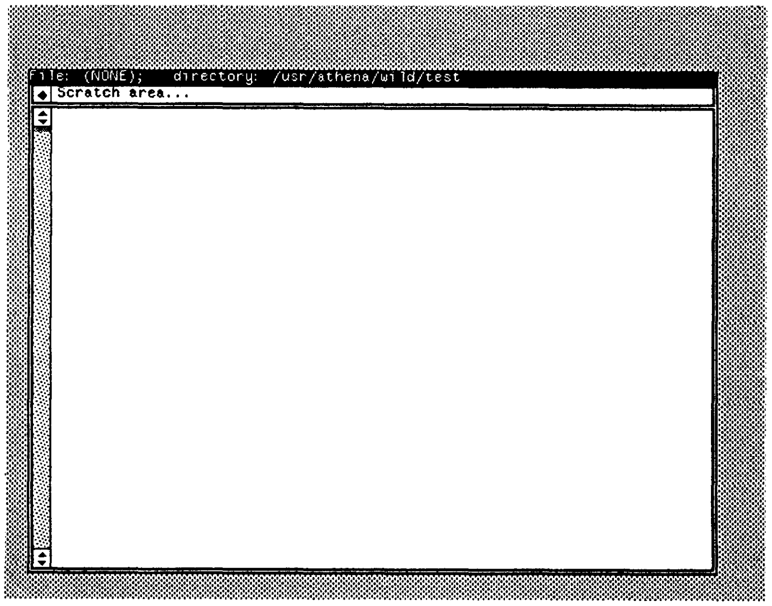
```
Shell Tool 3.0: /bin/csh
oscar%
```

`textedit`

A window and mouse text editing program, using the SunView text facility. See Chapter 8 .



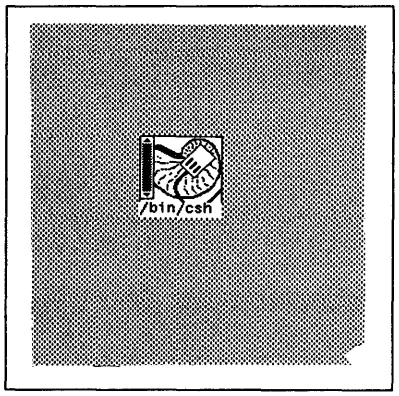
`textedit` icon



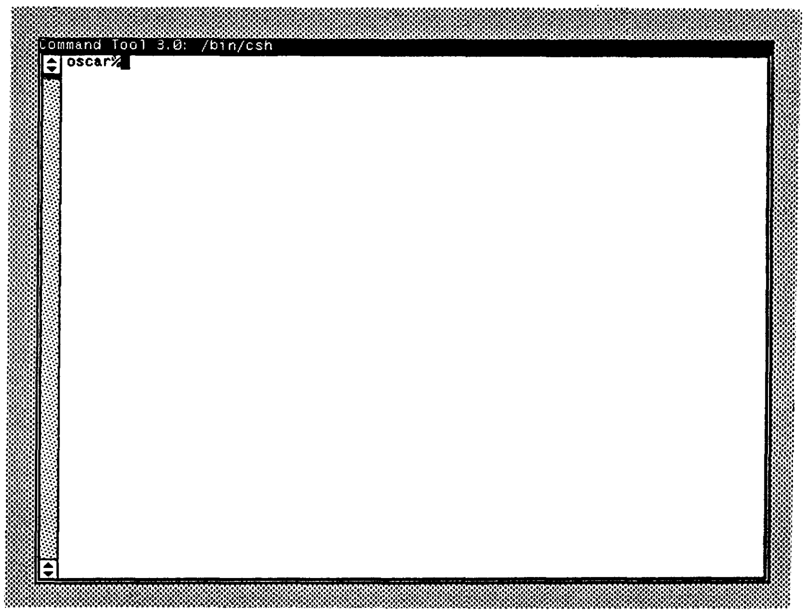
```
File: (NONE);    directory: /usr/athena/wild/test
Scratch area...
```

`cmdtool`

A `shelltool` combined with the conveniences of the SunView text facility. See Chapter 10.1 .



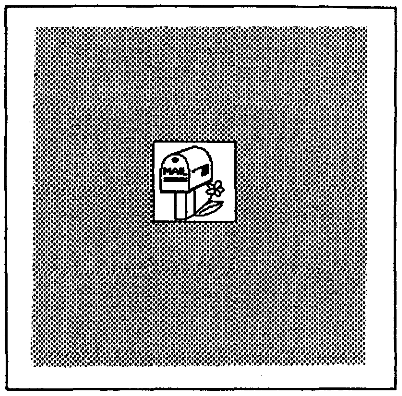
`cmdtool` icon



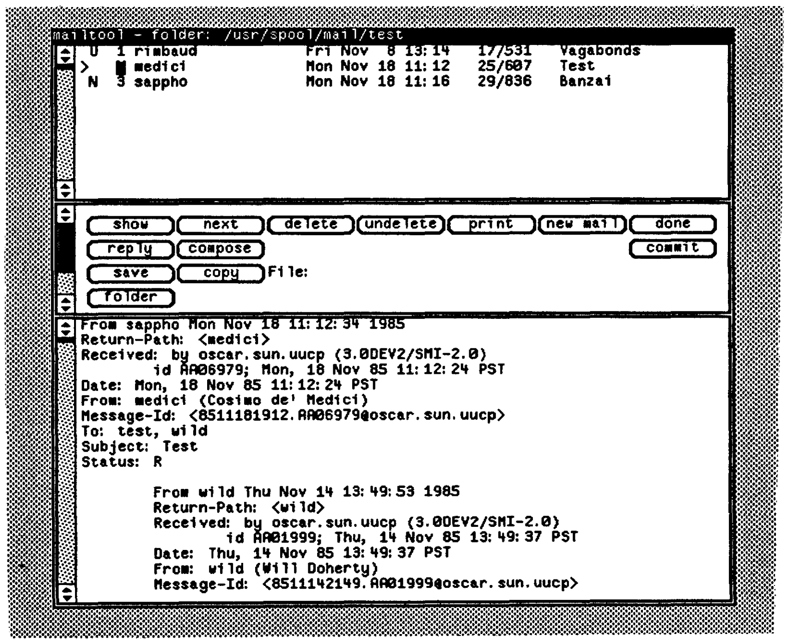
```
Command Tool 3.0: /bin/csh
oscar%
```

`mailtool`

A window and mouse mail program. See *Mail and Messages: Beginner's Guide.*



`mailtool` icon

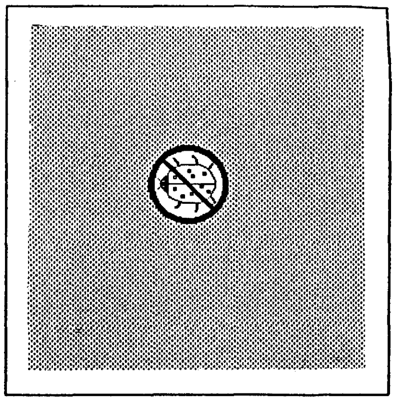


```
 U  1 rimbaud          Fri Nov  8 13:14   17/531   Vagabonds
>     medici           Mon Nov 18 11:12   25/607   Test
 N  3 sappho           Mon Nov 18 11:16   29/836   Banzai
```

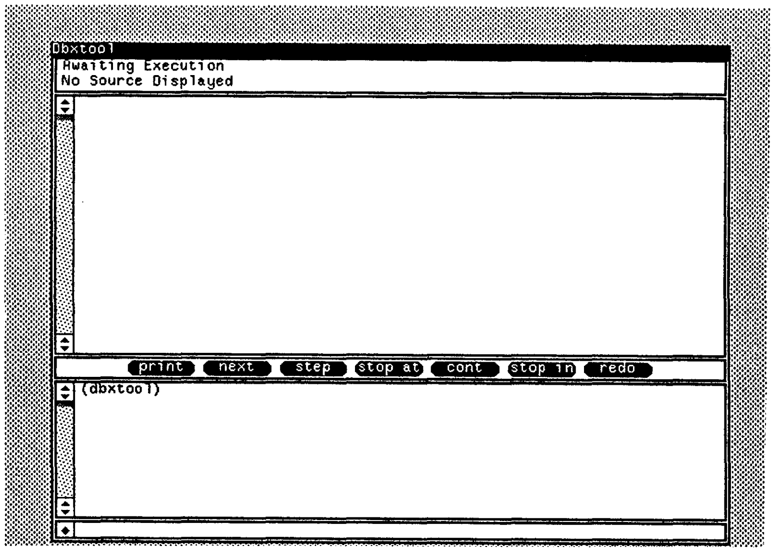
```
From sappho Mon Nov 18 11:12:34 1985
Return-Path: <medici>
Received: by oscar.sun.uucp (3.0DEV2/SMI-2.0)
        id AA06979; Mon, 18 Nov 85 11:12:24 PST
Date: Mon, 18 Nov 85 11:12:24 PST
From: medici (Cosimo de' Medici)
Message-Id: <8511181912.AA06979@oscar.sun.uucp>
To: test, wild
Subject: Test
Status: R

        From wild Thu Nov 14 13:49:53 1985
        Return-Path: <wild>
        Received: by oscar.sun.uucp (3.0DEV2/SMI-2.0)
                id AA01999; Thu, 14 Nov 85 13:49:37 PST
        Date: Thu, 14 Nov 85 13:49:37 PST
        From: wild (Will Doherty)
        Message-Id: <8511142149.AA01999@oscar.sun.uucp>
```

`dbxtool`

A window and mouse debugging tool, based on the `dbx` debugging program. See *Program Debugging Tools for the Sun Workstation.*



`dbxtool` icon

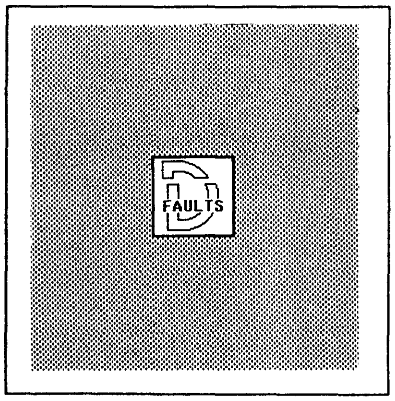


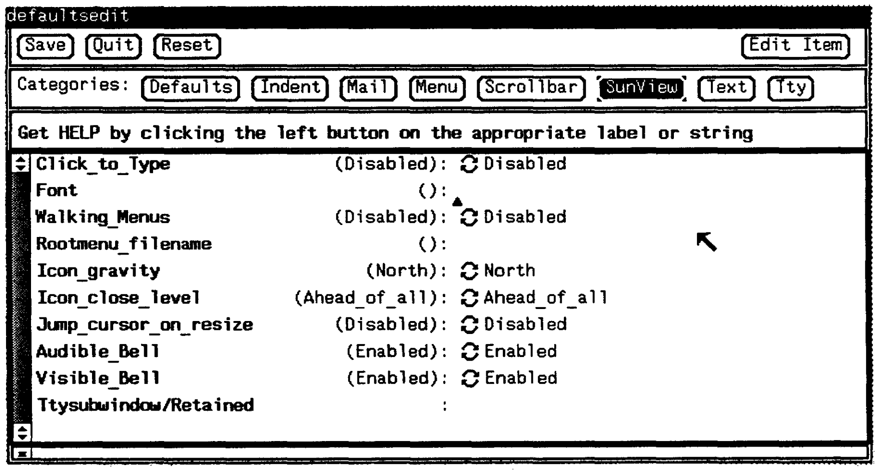
```
Awaiting Execution
No Source Displayed
```

```
(dbxtool)
```

`defaultsedit`

A window application that permits you to change the default parameters of the window system you run on your workstation. See Chapter 11 .

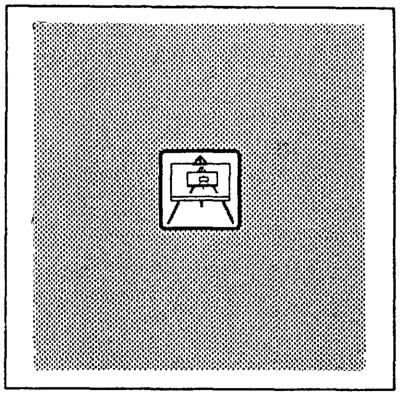

`defaultsedit` icon

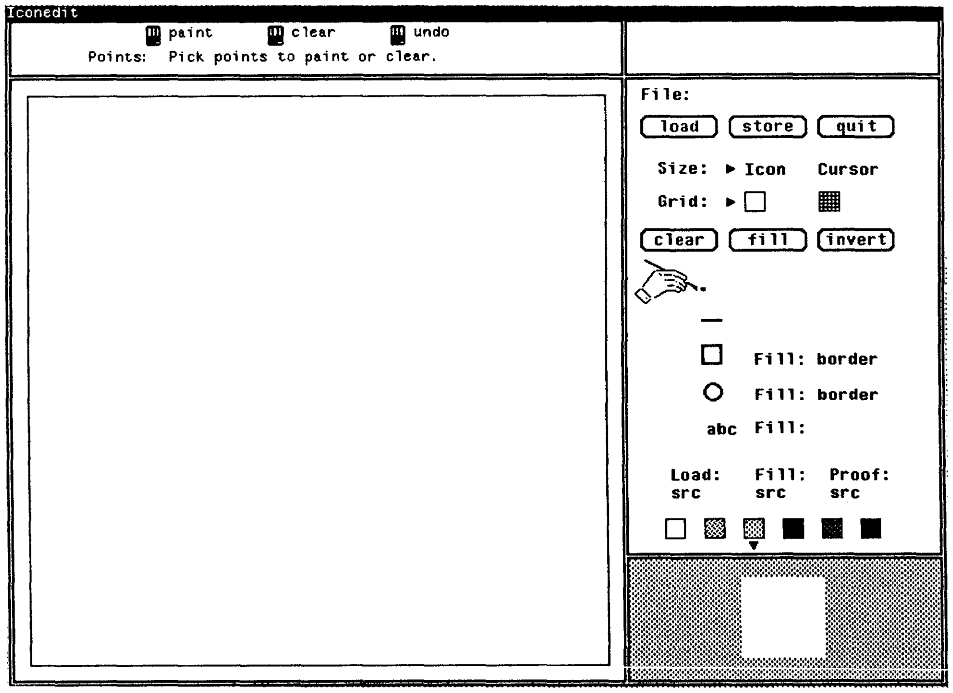


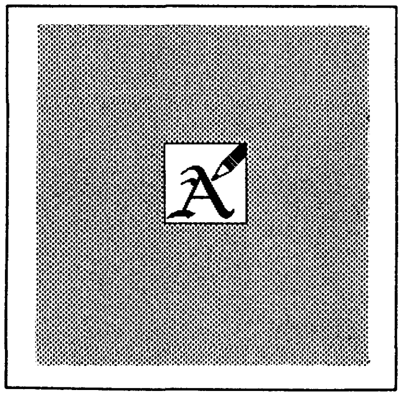
```
Get HELP by clicking the left button on the appropriate label or string
Click_to_Type                (Disabled): Disabled
Font                                (): 
Walking_Menus                (Disabled): Disabled
Rootmenu_filename                   ():
Icon_gravity                    (North): North
Icon_close_level         (Ahead_of_all): Ahead_of_all
Jump_cursor_on_resize        (Disabled): Disabled
Audible_Bell                  (Enabled): Enabled
Visible_Bell                  (Enabled): Enabled
Ttysubwindow/Retained                  :
```

`iconedit`

A tool that permits you to build your own window icons. See Chapter 11 .


`iconedit` icon

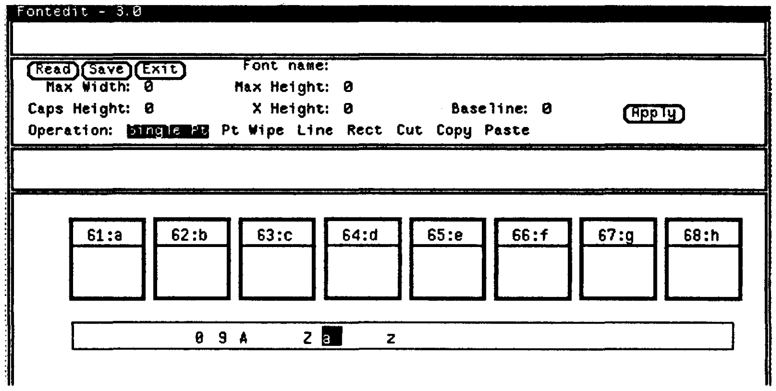




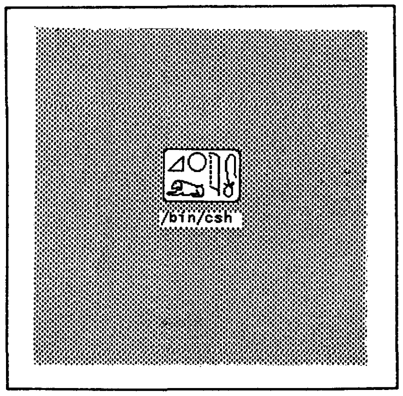
fontedit icon

`fontedit`

A tool that permits you to design your own window text fonts. See Chapter 11 .



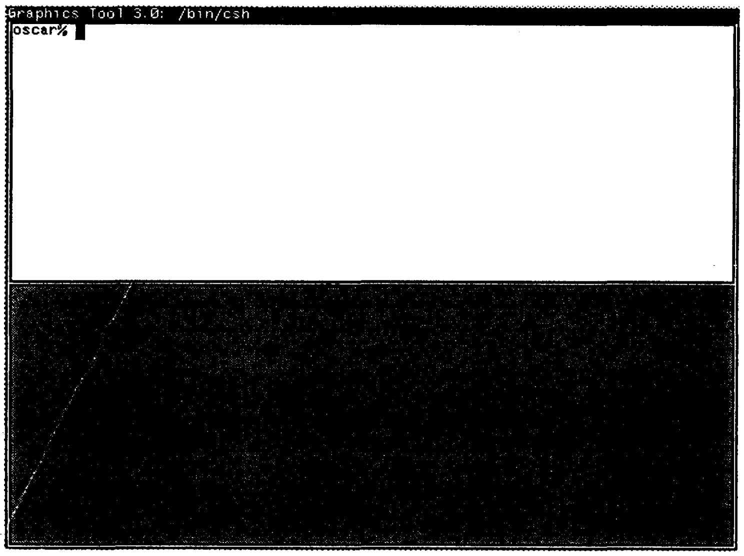
Font name:
Max Width: 0
Max Height: 0
Caps Height: 0
X Height: 0
Baseline: 0
Operation: Single Pt Pt Wipe Line Rect Cut Copy Paste



gfxtool icon

`gfxtool`

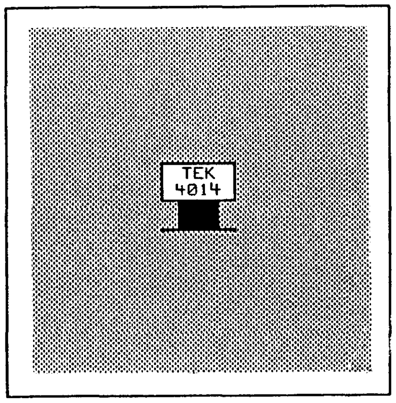
A tool that displays graphics programs. See Chapter 12 .



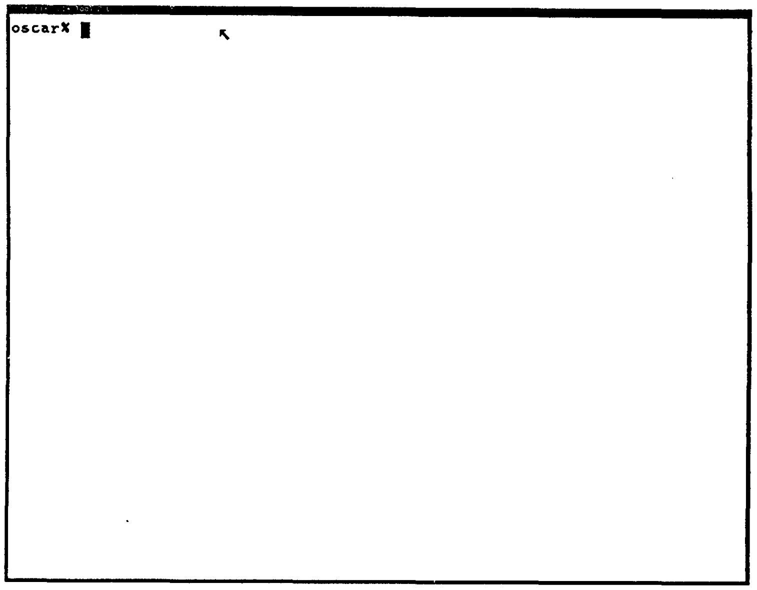
```
oscar%
```

`tektool`

A window that *emulates*, or acts like, a Tektronix 4014 ™ terminal. See Chapter 12 .



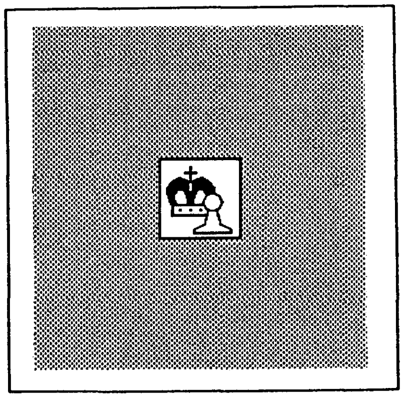
`tektool` icon



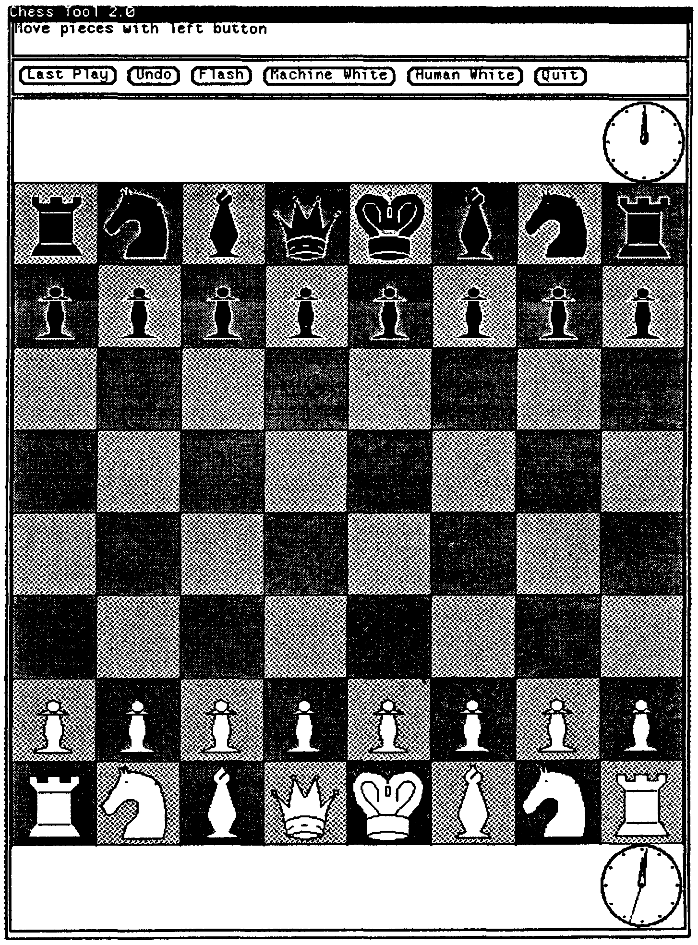
```
oscar%
```

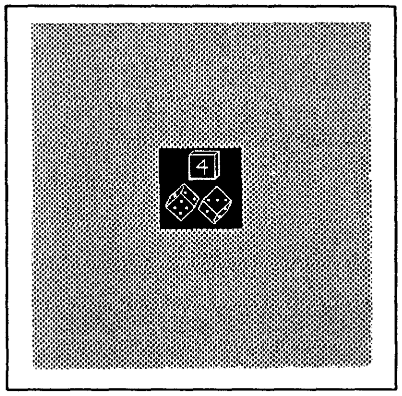
`chesstool`

A window-based chess game. See *Games, Demos, and Other Pursuits: Beginner's Guide.*



`chesstool` icon

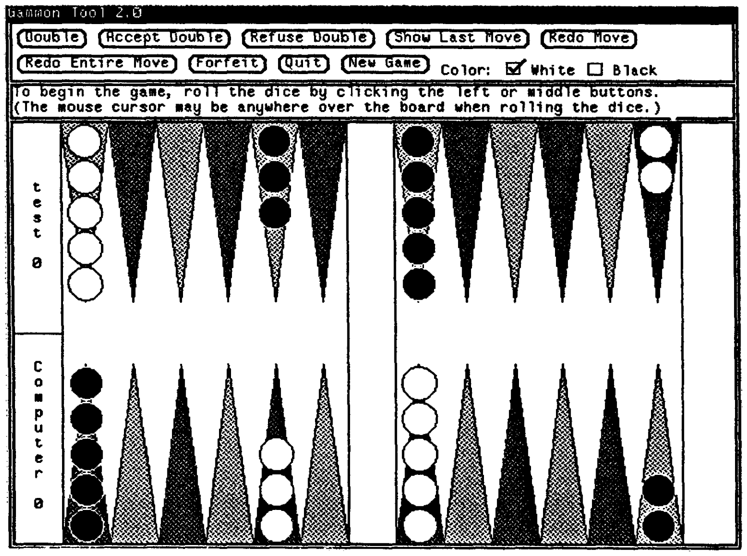




`gammontool` icon

`gammontool`

A window-based backgammon game. See *Games, Demos, and Other Pursuits: Beginner's Guide.*

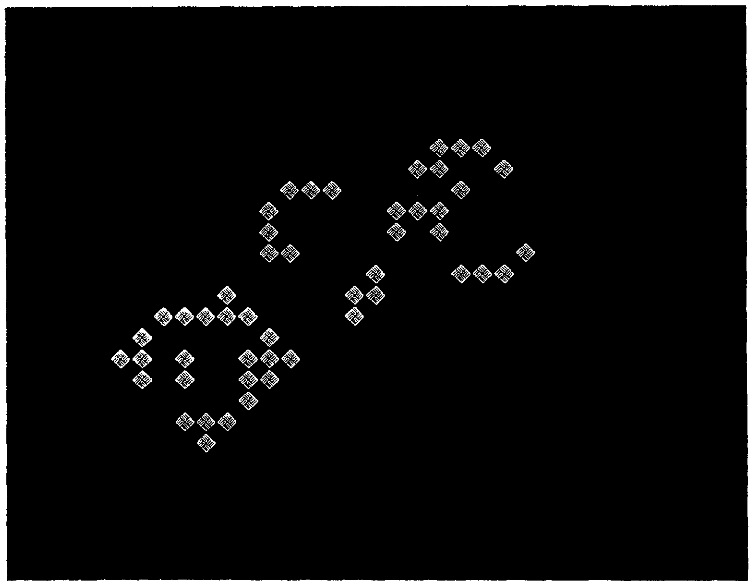


Color: [✓ White] ☐ Black

```
To begin the game, roll the dice by clicking the left or middle buttons.
(The mouse cursor may be anywhere over the board when rolling the dice.)
```

`lockscreen`

A program that "locks" the screen and displays the `life` graphics demo on it, until you give it your password to resume your work with the screen display as you left it before you ran `lockscreen`. See Section 17.1 in the chapter on other features of the window system. `lockscreen` has no icon.



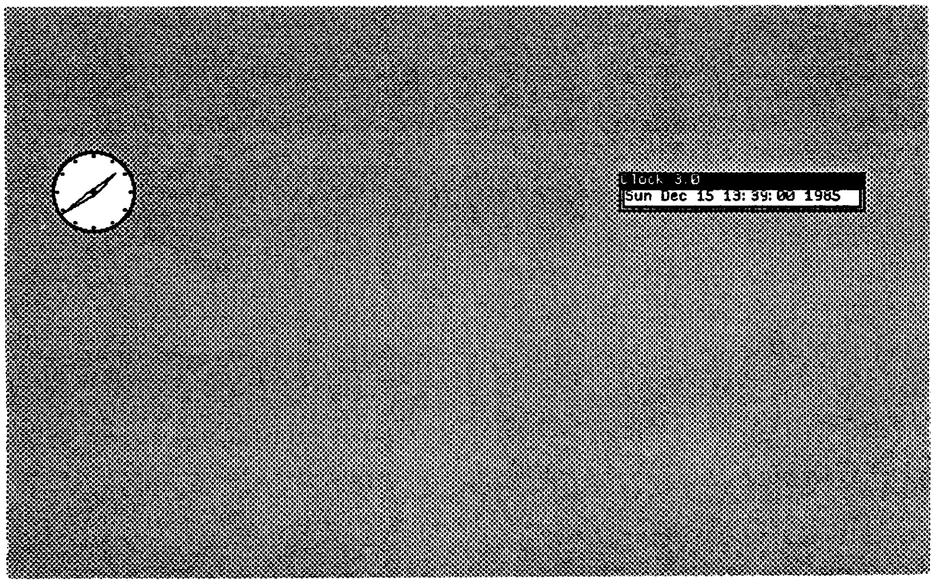
7.2. `clock`

`clock` displays the current date (day of the week, month, day of month, year) and time (hour, minute, and second) in the form of a digital clock when open, and an analog clock when closed.

To start a clock, type `clock &` to a `shelltool` command prompt. The ampersand character runs `clock` in the background of the `shelltool` where you started `clock`, so you can still type commands to that `shelltool`.

`clock` without any options looks like this:

Figure 7-1 `clock` *Without Any Options*



```
clock 3.0
Sun Dec 15 13:39:00 1985
```

`clock` has a number of options. For example when you want a clock that, while iconic, tells time in roman numerals with a "sweep" second hand and shows the date in the form day, date, and month, type `clock -S -r -d wdm`. For more information on `clock` and its options, see its Man Page, online or in the *Commands Reference Manual.*

7.3. `perfmeter`

`perfmeter`, or performance meter, provides a variety of statistics on the performance of the system. Simply choose the `PerfMeter` item on the root menu, or type `perfmeter` (followed by `&` when you want to run the tool in the background), to start up the tool, then use the frame menu to explore possibilities.

Stacking Menu

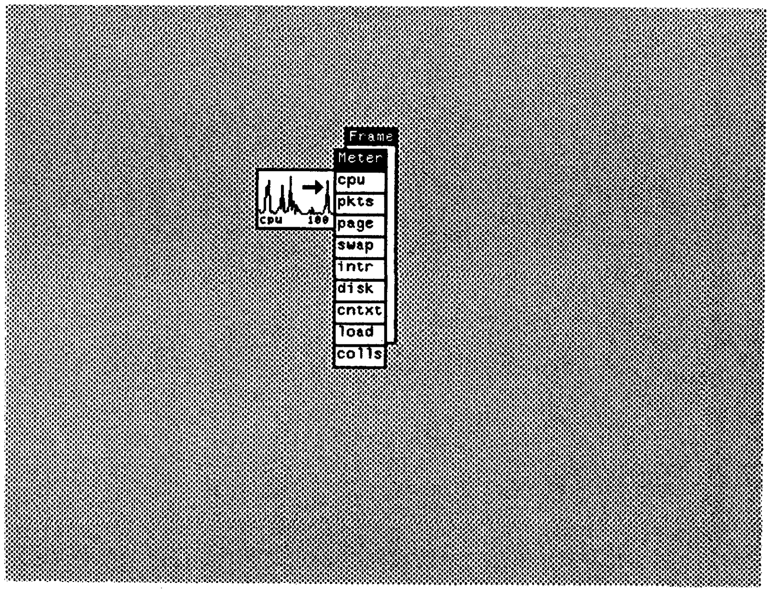
`perfmeter` has a *stacking menu*, an extension of the *old-style menu* you have seen so far. When you see a menu with overlapping parts, you can access any of the hidden menu items by:

- ▫ holding down the right mouse button
- ▫ moving the cursor onto the header of the hidden menu

- clicking the left mouse button

Then, continue to hold the right mouse button down, and move the cursor to choose items on the previously obscured menu portion.

Figure 7-2 `perfmeter` *and Stacking Frame Menu*



To find out more about the system statistics available, and about `perfmeter` in general, see the Man Page, online or in the *Commands Reference Manual.*

7.4. shelltool

`shelltool` is the window-based tool that you used prior to this chapter. `shelltool` runs the basic UNIX command interpreter, or shell, within a window, so you can type UNIX commands as you would when you aren't running `suntools`.

7.5. textedit

`textedit` is the primary file editing application for the SunView text facility. The next chapter describes `textedit` and the text facility.

8

Editing and the Text Facility

8

Editing and the Text Facility

This chapter is an introduction to the mouse-oriented, cut-and-paste *text facility*.[20]

Note: A *subwindow* is an area of the screen surrounded by a frame, or border and namestripe. More than one subwindow may fit inside a frame, although subwindows must be *tiled*; they can't overlap.

A *text subwindow* is a subwindow that provides display and editing capability for text within the text facility. The text facility permits various operations on text that appears in text subwindows, like *find* a character span or *undo* changes since the last selection.

A growing collection of applications are built on text subwindow capabilities, such as `textedit` for files, `cmdtool` for UNIX command interpretation, `mailtool` for mail operations, and `dbxtool` for program debugging.

Make sure to read the sections on setting up your system so you can use the *function keys* to invoke text facility operations, and using the new style of SunView menu called a *walking menu*, prior to reading the section in which you experiment with the text facility operations.

8.1. Setting Up Function Keys

Text facility *function keys* invoke text operations. The ten text facility function keys are: [Stop], [Again], [Props], [Undo], [Expose], [Put], [Open], [Get], [Find], and [Delete]. In addition, you could consider the [Caps Lock] key that you label as a function key.

You can locate the text facility *function keys* on one side or the other of your keyboard (except for the Sun-1 keyboard). To set up your function keys properly, find out what kind of keyboard you have: Sun-1, Sun-2, or Sun-3.

If, like most people, you have a Sun-2 or Sun-3 keyboard and want to move the mouse with your right hand, then you probably don't have to change anything. Right-handed mouse users with Sun-2 or Sun-3 keyboards use the keys [L1] through [L10], located on the left of the keyboard, as their function keys.

However, with a Sun-2 or Sun-3 keyboard, you can specify on which side of the keyboard you want the function keys using a program called `setkeys`. Right-handed people usually prefer to have function keys on the left side of the keyboard, because they can wield their mice on the right, while typing editing

[20] The text facility programming interface documentation appears in the chapter on text subwindows in the *SunView Programmer's Guide*.

function keys on the left. ''Left-moused'' people, however, prefer to wiggle mice on the left, typing function keys on the right (using R1, R4, R7, R10, R13, R3, R6, R9, R12, and R15).

Sun-1 keyboards are the keyboards distributed with the Sun-100u workstation. Sun-1 keyboards only have function keys on the right side of the keyboard.

Once you find out what kind of keyboard you have, and you choose which side of the keyboard to put the function keys on, you can specify this choice to the system using `setkeys`. The following table shows what to put in your `.login` file to set up your function keys.

Note: When you type `lefty` as an argument to `setkeys`, LINE FEED acts just like CTRL, but saves a lot of finger-stretching because it's closer to the function keys on the right side of the keyboard.

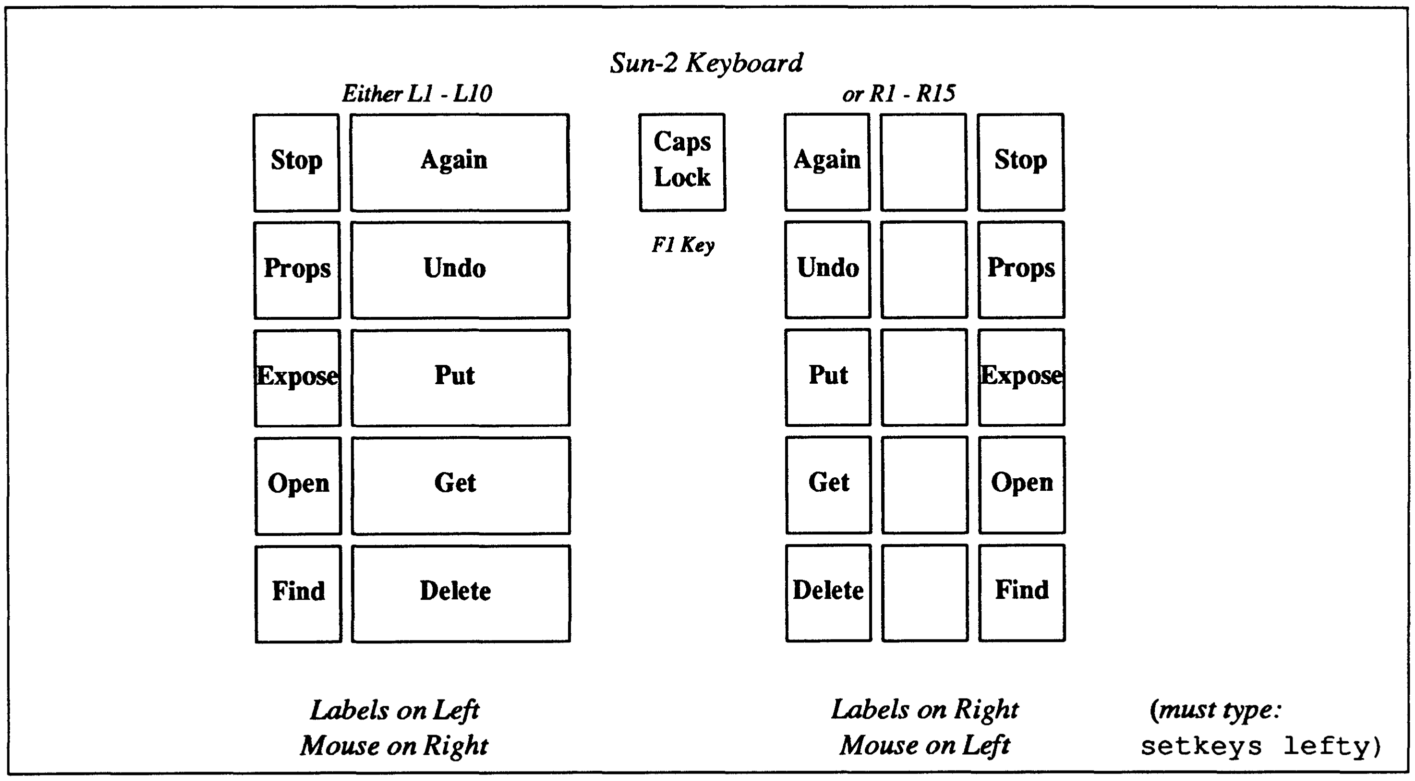
Table 8-1 *Setting Up Your Function Keys*

Keyboard Type	*Keyboard Side*	*Typical Handedness*	*Text in* `.login` *File*
`Sun-1`	`Right only`	`Left or Right`	`setkeys sun1`
`Sun-2`	`Left`	`Right`	`setkeys sun2` *(Defaul*
`Sun-2`	`Right`	`Left`	`setkeys sun2 lefty`
`Sun-3`	`Left`	`Right`	`setkeys sun3` *(Defaul*
`Sun-3`	`Right`	`Left`	`setkeys sun3 lefty`

Note: Put the cursor over the gray screen background as you affix the labels to the keys, so you don't execute a text operation accidentally.

After you have set up your function keys, affix the appropriately sized and marked labels, located within the back cover of this document, over the indicated keys for your keyboard type.

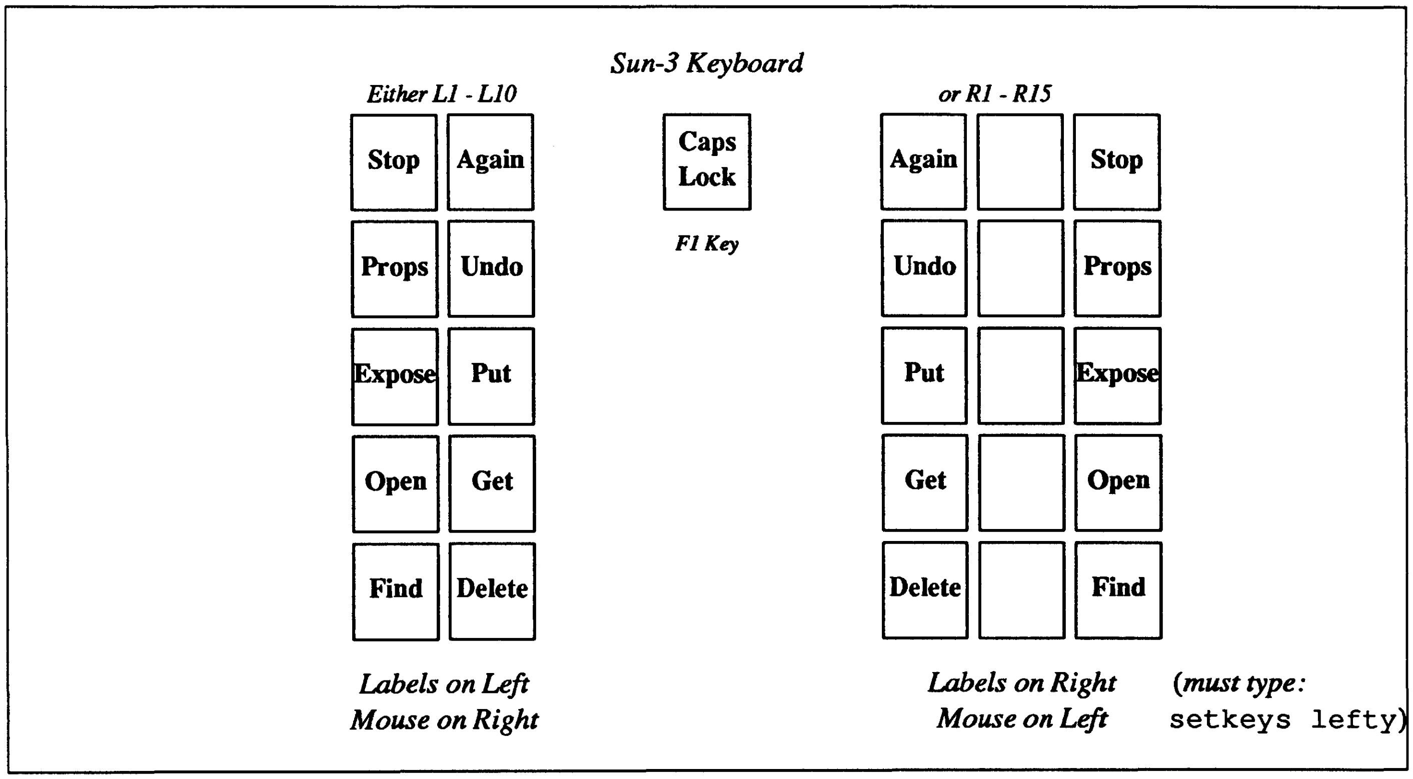
Figure 8-1 *Affix Labels on Left or Right of Sun-2 Keyboard*



The even-numbered keys between [L1] and [L10] are thinner on the Sun-3 keyboard than on the Sun-2 keyboard, so use the thin labels for those function keys.

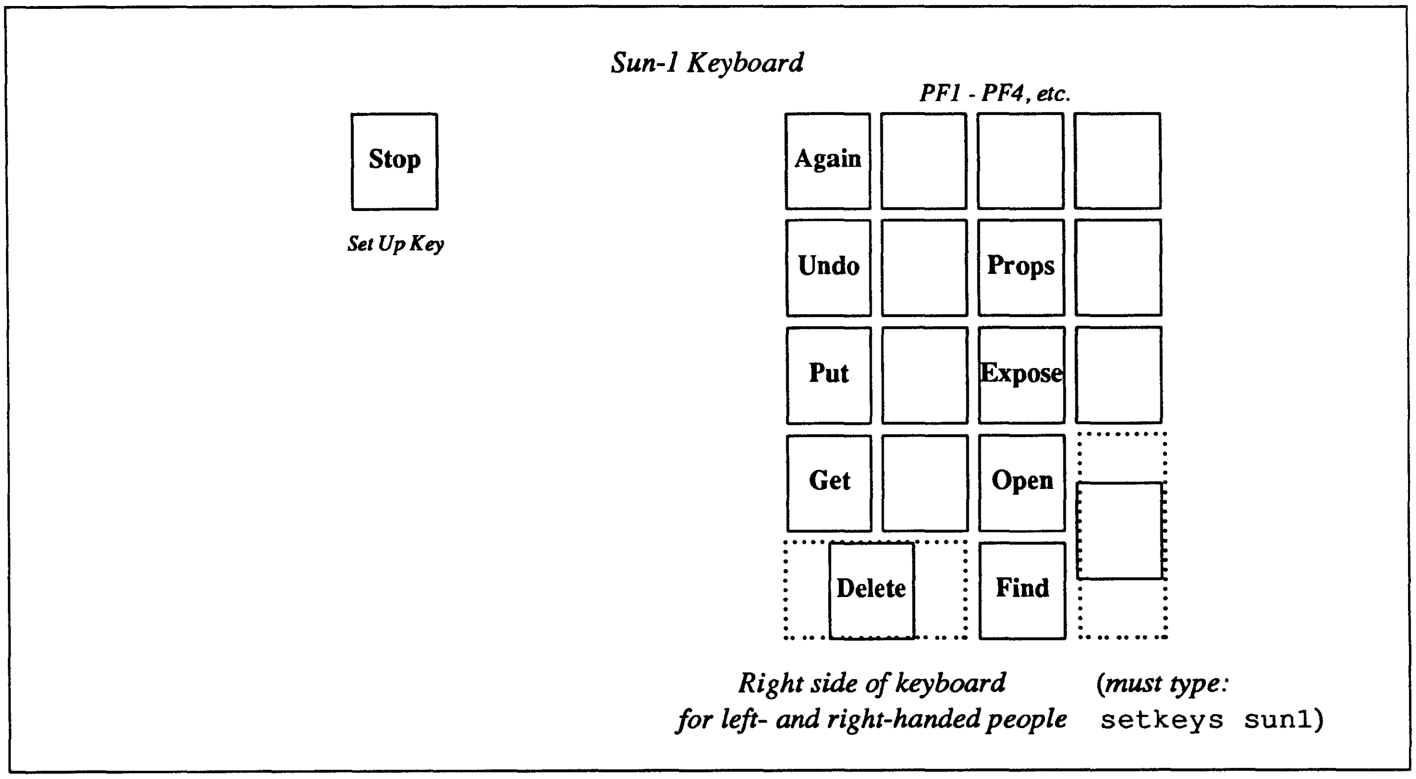
Sun-3 keyboards also have a pre-labeled [Caps Lock] key, but the [F1] keys acts as a [Caps Lock] key for individual windows. You probably want to affix the [Caps Lock] label on the [F1] key anyway.

Figure 8-2 *Affix Labels on Left or Right of Sun-3 Keyboard*



The Sun-1 keyboard only supports text facility function keys on the right side of the keyboard; there are no keys on the left side. Because there is a predefined [Caps Lock] key on the Sun-1 keyboard and no [F1] key to act as an additional [Caps Lock], you don't need the [Caps Lock] label for the Sun-1 keyboard. [Stop], however, appears on the left side, so affix the [Stop] label on the [SET UP] key.

Figure 8-3 *Affix Labels on Right Side of Sun-1 Keyboard*



To find out more about the arrow keys, resetting the function key set up, and `setkeys` in general, see the `setkeys` Man Page, online or in the *Commands Reference Manual.* If necessary, you should be able to remove and restick the labels on new function keys without damaging the labels or the keyboard.

If you have a problem with arrow keys or function keys, such as the:

```
.ttyswrc error: keyname cannot be mapped.
```

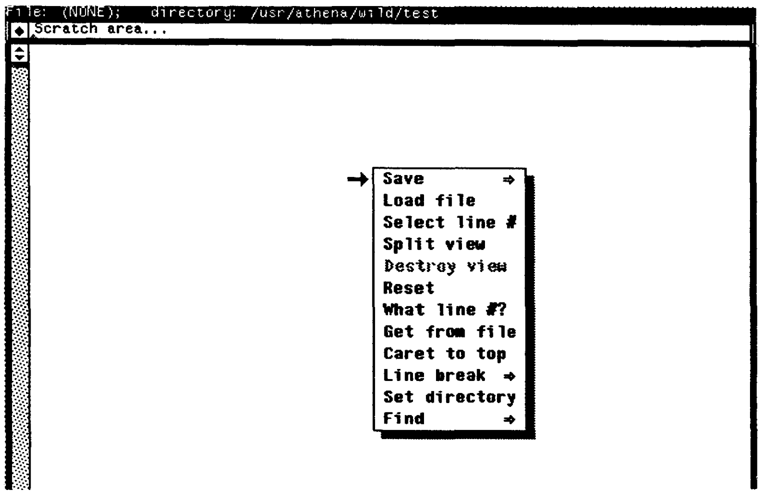
notification when starting up a subwindow, try looking in the chapter about modifying subwindow behavior, Chapter 16 .

8.2. New-Style Menus, or Walking Menus

So far, you have probably used only *old-style menus* and *stacking menus.* The frame and root menus appear in the old style unless you specify otherwise with `defaultsedit` as discussed at the end of this section. In the description of `perfmeter` (Section 7.3), you learned about stacking menus.

Walking menus are similar to stacking menus, but you can more easily pick hidden menu items with walking menus. A walking menu looks like an old-style menu, except that some menu items have an arrow that points to the right (=>). You can move the mouse through these arrows to make a hidden *pull-right* menu appear, so you can pick additional menu items.

Figure 8-4 *Typical Walking Menu (for* `textedit`*)*



Step-by-step instructions for choosing hidden menu items on the pull-right portions of walking menus at the end of the section on basic editing with `textedit` (Section 8.3).

Since new-style walking menus are generally easier to use than the old-style menus, you should probably switch to the walking menu style using `defaultsedit` in category `SunView` to set the attribute `Walking_Menus` to `Enabled`.[21]

8.3. Basic Editing

This section explains briefly how to edit text in text subwindows, including these operations on text:

- *insert*
- *delete*

[21] See the section on `defaultsedit` (Section 11.1).

- *copy*
- *move*
- *replace*

In addition, you learn how to *load* and *save* a file with the text editor `textedit`. The next chapter (Chapter 9) describes more advanced editing techniques.

Terminology

Learning the following terms makes it easier to read the rest of this manual:

Note: *Inverse video* means reversed light and dark portions of the screen.

Note: Within the text facility, [DEL], the delete character key, is *completely different* from [Delete], the delete function key.

[LEFT]	Left button on mouse.
[MIDDLE]	Middle button on mouse.
[RIGHT]	Right button on mouse.
cursor	The distinctive shape, usually an arrow pointing ''north-west,'' that follows and indicates the position of the mouse.
point at	Position the cursor over.
click	Press and release a mouse button.
select	Indicate a *span* (a contiguous sequence) of characters by moving the cursor and clicking.
selection	A span of characters, highlighted by inverse video, underlining, or gray shading.
primary selection	Or *current selection*, span of text highlighted by inverse video, and marking the area for current editing operations.
adjusting a selection	Moving the cursor and clicking to make a selection encompass more or fewer characters.
insertion point	The place where the next character you insert will go.
caret	A blinking triangular, gray diamond, or rectangular solid or outline shape that indicates the insertion point.
shelf	A place where the text facility stores the text you select or delete, using the [Put] or [Delete] function keys.
scrolling	Moving the text up or down in a subwindow; can also think of it as repositioning the subwindow with respect to the data it displays.
scroll bar	A gray column at the left of the text subwindow used for scrolling and indicating position within the entire text.
scroll button	The box at the top or bottom of the scroll bar used for scrolling.

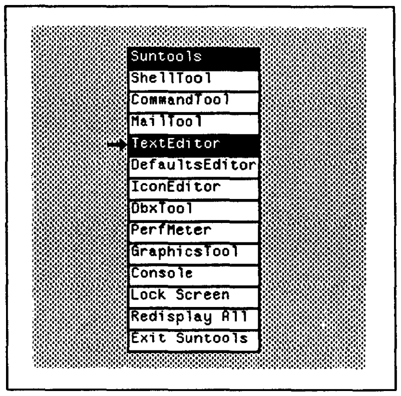
menu	A list of operations that you can choose by pressing and holding RIGHT and moving the cursor. Menus appear when, and for as long as, you hold down RIGHT. The operation occurs when you release RIGHT with the cursor over a menu item.
menu item	The area of a menu that you pick using the mouse to invoke a particular operation.
root menu	The menu you obtain by holding down RIGHT with the cursor over the gray background area of the screen.
frame menu	The menu you obtain by holding down RIGHT over the border or namestripe of any window. Contains the operations `Close`, `Move`, `Resize`, `Expose`, `Hide`, `Redisplay`, and `Quit`.
text subwindow	A subwindow that provides display and editing capability for text.
command subwindow	A text subwindow that includes a command interpreter, like `cmdtool`.
tty subwindow	A subwindow with a command interpreter that doesn't support some of the text facility editing operations, like `shelltool`.
text menu	The menu you obtain by holding down RIGHT anywhere inside a text subwindow. Contains operations such as `Save`, `Load File`, `Reset`, and `Find`.

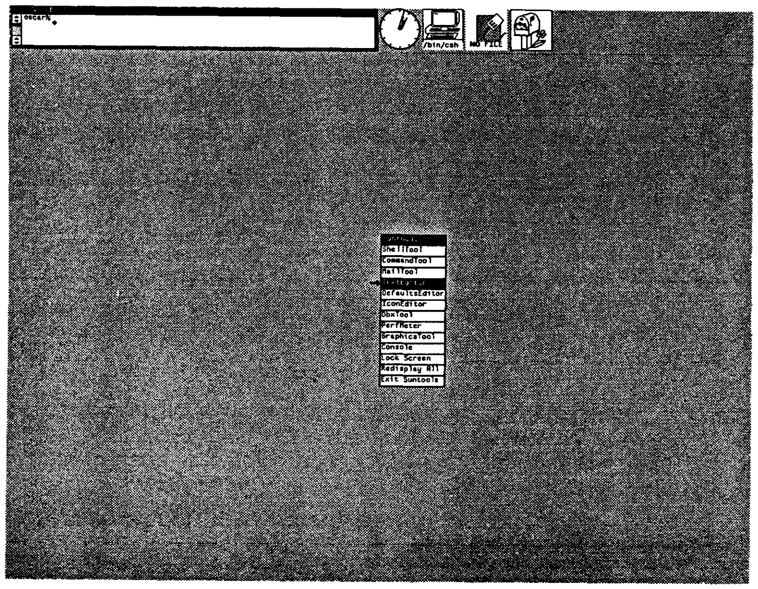
Starting `textedit`

To edit or examine a file, pop up a root menu and choose the `TextEditor` item to start up a `textedit` text subwindow. After a moment, the `textedit` appears on your screen.[22]

[22] If you have your own root menu, see Chapter 15 on modifying root menus, you may want to add to it a line which invokes `textedit`.

Figure 8-5 *Starting* `textedit` *With the* `TextEditor` *Root Menu Item*





As an alternative to selecting the `TextEditor` root menu item, you can type `textedit`, followed by the ampersand character (`&`) for background processing, to a `shelltool` command prompt.

Figure 8-6 *Starting* `textedit` *From* `shelltool` *Command Prompt*

```
venus% textedit &
venus%
```

Loading the Initial File

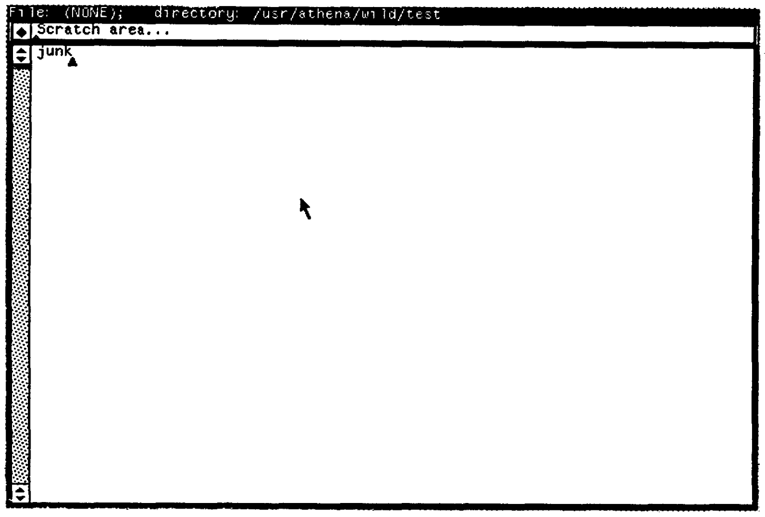
Note: Whenever the text subwindow is empty, you can type filename in it and press ESC to load a file. An empty text subwindow displays `File: (NONE)` in its namestripe; select the `Reset` text menu item once or twice to make any text subwindow empty.

Move the cursor into the `textedit` window and type the filename of the file you wish to edit, followed by ESC.

Choose a file that you don't mind damaging while you experiment with the text editor. One possibility is to copy your `.login` to a file called `junk` (using the `cp` command in a `shelltool`), then load `junk` into `textedit` — you can remove `junk` when you're finished experimenting with `textedit`.

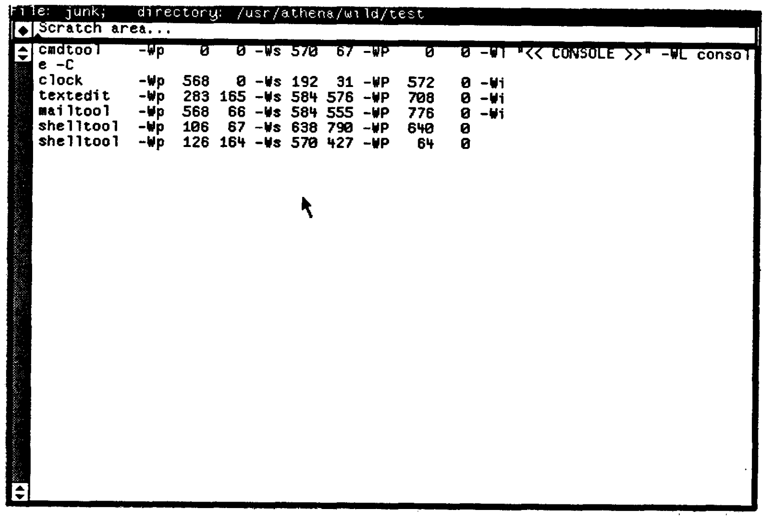
Figure 8-7 *Loading the Initial File into* `textedit`*: Before*

Note: When the `textedit` window is iconic, the icon displays the name of the file currently loaded in `textedit`.



```
File: (NONE);    directory: /usr/athena/wild/test
Scratch area...
junk
```

Figure 8-8 *Loading the Initial File into* `textedit`*: After*



```
File: junk;    directory: /usr/athena/wild/test
Scratch area...
cmdtool    -Wp    0   0 -Ws 570  67 -WP    0   0 -Wl "<< CONSOLE >>" -WL consol
e -C
clock      -Wp  568   0 -Ws 192  31 -WP  572   0 -Wi
textedit   -Wp  283 165 -Ws 584 576 -WP  708   0 -Wi
mailtool   -Wp  568  66 -Ws 584 555 -WP  776   0 -Wi
shelltool  -Wp  106  67 -Ws 638 790 -WP  640   0
shelltool  -Wp  126 164 -Ws 570 427 -WP   64   0
```

When you start `textedit` from the command line in a `shelltool`, `textedit`'s working directory starts as the current directory of the `shelltool`.

But when you create a `textedit` window using the root menu, `textedit`'s working directory is the directory from which you started `suntools`, most often your home directory.

For information about changing directories, see the section on changing directories later in this chapter.

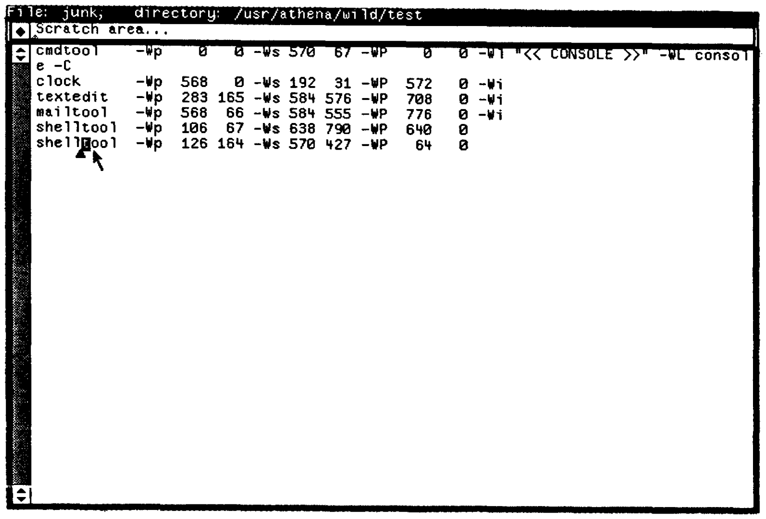
Selecting Text

You need to know how to specify text selections as operands for text menu editing operations, like loading a file and storing to a file. Text selections also act as operands for function key editing operations, such as *delete* and *find*. So you need to learn how to select portions of text efficiently.

Selecting a Character

Select a single character by moving the cursor over the character and clicking [LEFT]. `textedit` highlights the selected character using inverse video.

Figure 8-9 *Selecting a Character*



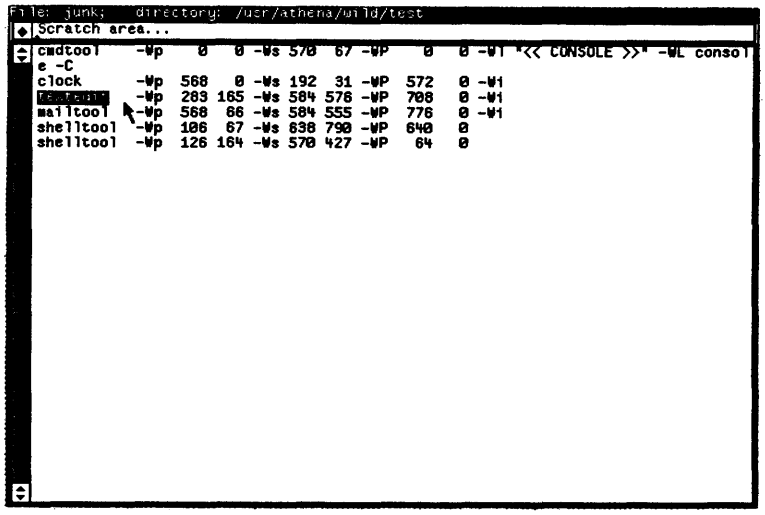
```
File: junk;    directory: /usr/athena/wild/test
Scratch area...
cmdtool     -Wp    0   0 -Ws 570  67 -WP    0   0 -Wl "<< CONSOLE >>" -WL consol
e -C
clock       -Wp  568   0 -Ws 192  31 -WP  572   0 -Wi
textedit    -Wp  283 165 -Ws 584 576 -WP  708   0 -Wi
mailtool    -Wp  568  66 -Ws 584 555 -WP  776   0 -Wi
shelltool   -Wp  106  67 -Ws 638 790 -WP  640   0
shelltool   -Wp  126 164 -Ws 570 427 -WP   64   0
```

Alternatively, you can hold down [LEFT] and move the cursor to the correct place before releasing the mouse button.

Selecting a Span of Characters

Select a *span* of characters by selecting the the first or last character (click [LEFT] over it), then *adjust* the selection by pointing at the other endpoint and clicking [MIDDLE].

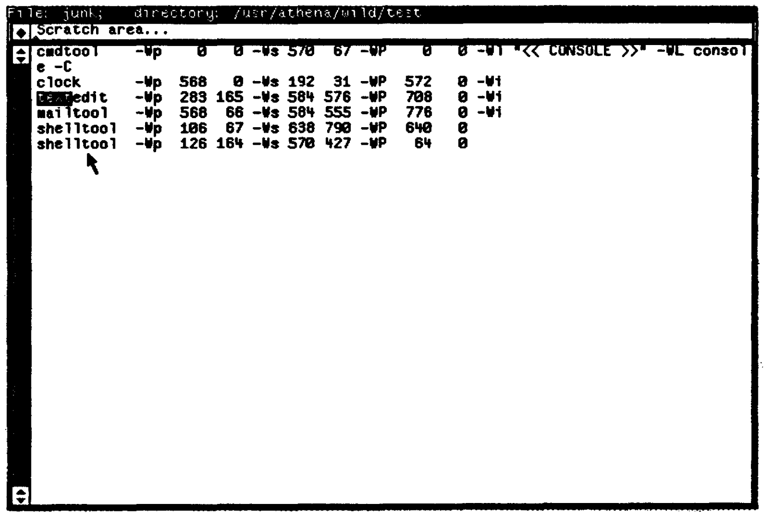
Figure 8-10 *Selecting a Character Span*



```
File: junk;    directory: /usr/athena/wild/test
Scratch area...
cmdtool    -Wp    0    0 -Ws 570  67 -WP    0    0 -Wl "<< CONSOLE >>" -WL consol
e -C
clock      -Wp  568    0 -Ws 192  31 -WP  572    0 -Wi
textedit   -Wp  283  165 -Ws 584 576 -WP  708    0 -Wi
mailtool   -Wp  568   66 -Ws 584 555 -WP  776    0 -Wi
shelltool  -Wp  106   67 -Ws 638 790 -WP  640    0
shelltool  -Wp  126  164 -Ws 570 427 -WP   64    0
```

You can also hold [MIDDLE] down and move the endpoint to a new position before releasing the mouse button.

Figure 8-11 *Adjusting a Selection*



```
File: junk;    directory: /usr/athena/wild/test
Scratch area...
cmdtool    -Wp    0    0 -Ws 570  67 -WP    0    0 -Wl "<< CONSOLE >>" -WL consol
e -C
clock      -Wp  568    0 -Ws 192  31 -WP  572    0 -Wi
textedit   -Wp  283  165 -Ws 584 576 -WP  708    0 -Wi
mailtool   -Wp  568   66 -Ws 584 555 -WP  776    0 -Wi
shelltool  -Wp  106   67 -Ws 638 790 -WP  640    0
shelltool  -Wp  126  164 -Ws 570 427 -WP   64    0
```

You can adjust a selection as many times as you like, and you can adjust either end of the selection; you adjust the end of the selection closest to the cursor when you press [MIDDLE]. You can make a selection larger *or* smaller when adjusting it.

For more advanced selection techniques, see the chapter on advanced editing, Chapter 9 .

Insertion Point

When you click the left or middle button on the mouse, you actually do two things:

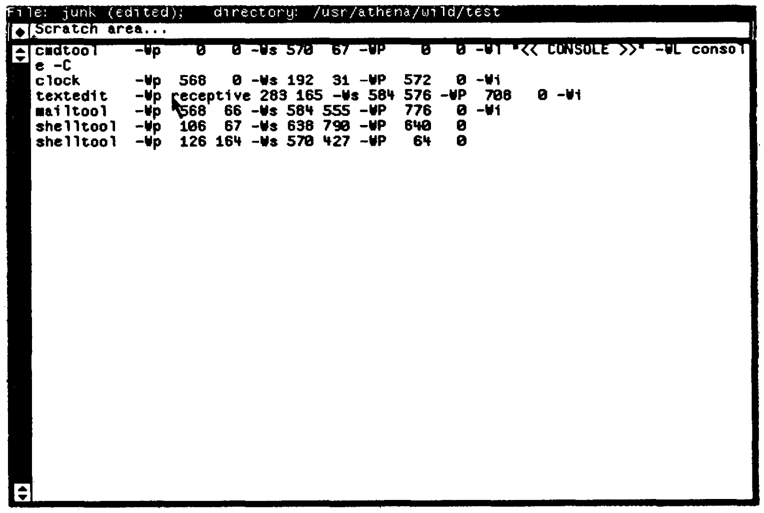
- Select a character or characters
- Specify the *insertion point.*

Note: As soon as you edit the text in a `textedit` text subwindow, the character string `(edited)` appears after the filename in the namestripe. `textedit`, while iconic, displays a greater than symbol (>) in front of the filename that you've edited, but not yet saved.

The insertion point is the place where the next character you insert will go.

Try clicking LEFT at some spot in the file to set the insertion point, then type in some text.

Figure 8-12 *Typing Text at the Insertion Point*

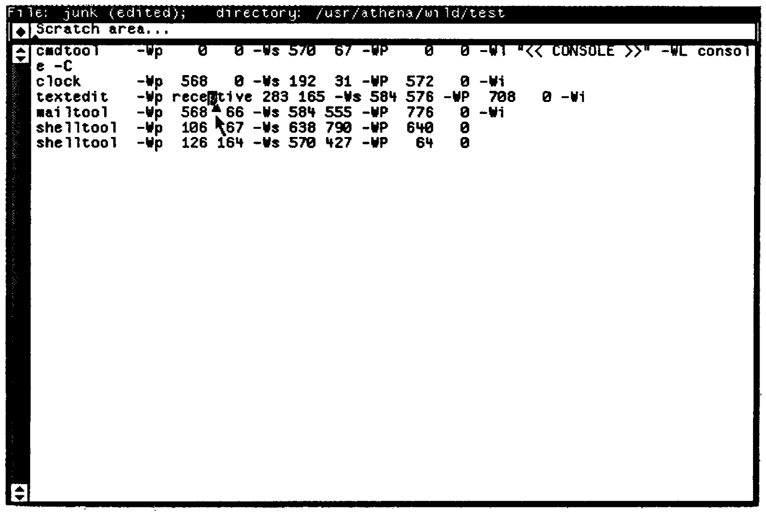


```
File: junk (edited);    directory: /usr/athena/wild/test
Scratch area...
cmdtool     -Wp    0   0 -Ws 570  67 -WP    0   0 -Wi "<< CONSOLE >>" -WL consol
e -C
clock       -Wp  568   0 -Ws 192  31 -WP  572   0 -Wi
textedit    -Wp receptive 283 165 -Ws 584 576 -WP  708   0 -Wi
mailtool    -Wp  568  66 -Ws 584 555 -WP  776   0 -Wi
shelltool   -Wp  106  67 -Ws 638 790 -WP  640   0
shelltool   -Wp  126 164 -Ws 570 427 -WP   64   0
```

Whenever you click the mouse in a text subwindow, you highlight one or more characters in inverse video and the *caret*, a blinking triangle, appears at the end of the selection closest to where the cursor was.

For example, to place the insertion point between letters `p` and `t` in character string `receptive`, position the cursor between `p` and `t` (actually anywhere between the middle of `p` and the middle of `t` will do) and click LEFT.

Figure 8-13 *Moving the Insertion Point*



```
File: junk (edited);    directory: /usr/athena/wild/test
Scratch area...
cmdtool     -Wp    0    0 -Ws 570  67 -WP    0    0 -Wi "<< CONSOLE >>" -WL consol
e -C
clock       -Wp  568    0 -Ws 192  31 -WP  572    0 -Wi
textedit    -Wp receptive 283 165 -Ws 584 576 -WP  708    0 -Wi
mailtool    -Wp  568   66 -Ws 584 555 -WP  776    0 -Wi
shelltool   -Wp  106  167 -Ws 638 790 -WP  640    0
shelltool   -Wp  126  164 -Ws 570 427 -WP   64    0
```

Besides moving the insertion point, you select the character closest to where you placed the cursor, in this case `p` or `t`. But in the above example, the user clicked slightly closer to the left side of the insertion point, selecting `p`.

The insertion point is always *between* characters, eliminating the need to specify whether you are inserting before or appending after a particular character.

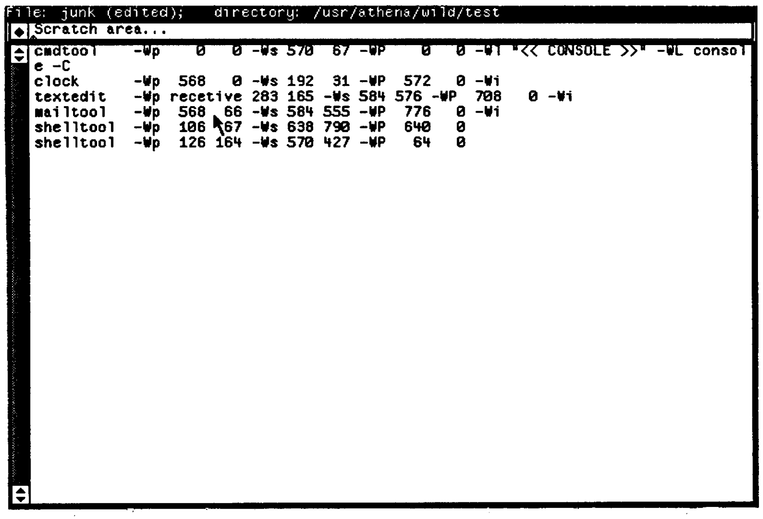
Basic Editing Operations

Summing up insertion of text — to insert characters in a text subwindow, simply place the caret where you want the characters to go and start typing. There is no insert command; unlike `vi`, the editor is *modeless*.

Deleting the Character to the Left of the Caret

You can delete the character immediately to the left of the caret using your normal UNIX rubout character (DEL , the delete key, or BACKSPACE for most users).

Figure 8-14 *Deleting the Character to the Left of the Caret*

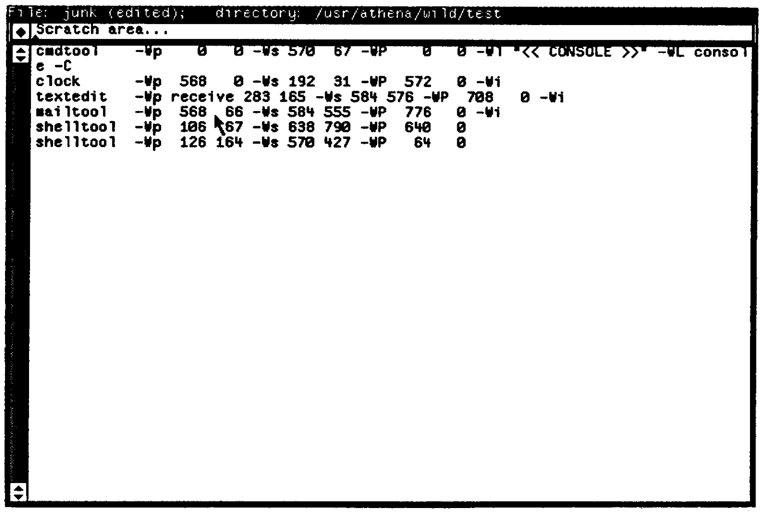


```
File:  junk (edited);    directory:  /usr/athena/wild/test
Scratch area...
cmdtool     -Wp     0    0 -Ws 570  67 -WP     0    0 -Wl "<< CONSOLE >>" -WL consol
e -C
clock       -Wp   568    0 -Ws 192  31 -WP   572    0 -Wi
textedit    -Wp recetive 283 165 -Ws 584 576 -WP  708   0 -Wi
mailtool    -Wp   568   66 -Ws 584 555 -WP   776    0 -Wi
shelltool   -Wp   106  67 -Ws 638 790 -WP   640    0
shelltool   -Wp   126 164 -Ws 570 427 -WP    64    0
```

Deleting the Character to the Right of the Caret

You can also delete the character immediately to the *right* of the caret by holding the SHIFT key down and typing the rubout character. This ability to reverse the direction of the delete illustrates the general "toggling" property of the SHIFT key, especially when used in conjunction with text facility function keys.

Figure 8-15 *Deleting the Character to the Right of the Caret*

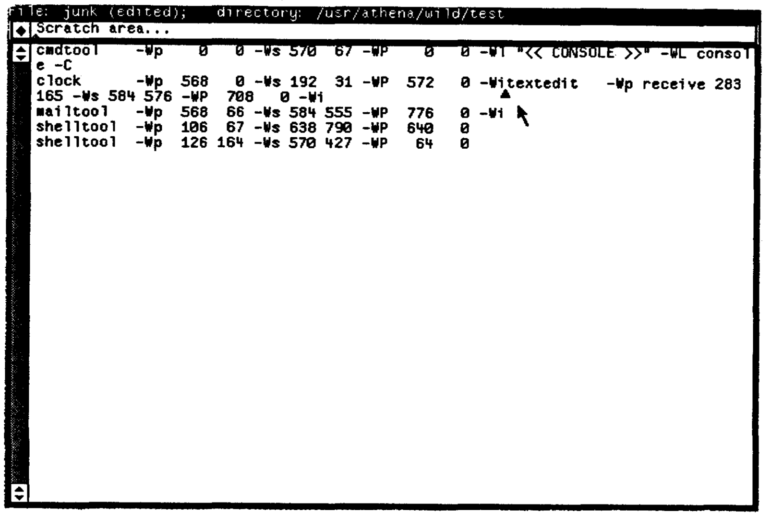


```
File:  junk (edited);    directory:  /usr/athena/wild/test
Scratch area...
cmdtool     -Wp     0    0 -Ws 570  67 -WP     0    0 -Wl "<< CONSOLE >>" -WL consol
e -C
clock       -Wp   568    0 -Ws 192  31 -WP   572    0 -Wi
textedit    -Wp receive 283 165 -Ws 584 576 -WP  708   0 -Wi
mailtool    -Wp   568   66 -Ws 584 555 -WP   776    0 -Wi
shelltool   -Wp   106  67 -Ws 638 790 -WP   640    0
shelltool   -Wp   126 164 -Ws 570 427 -WP    64    0
```

Note that it doesn't matter whether you typed the character as part of this insertion, or it was already there from before this editing session, you can still delete it.

When you delete a newline character, you join the two lines into one.

Figure 8-16 *Deleting a Line Break*

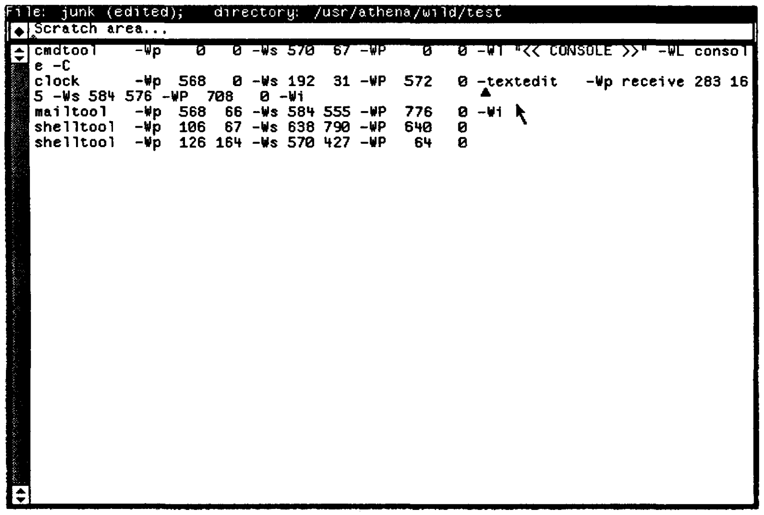


```
File: junk (edited);   directory: /usr/athena/wild/test
Scratch area...
cmdtool    -Wp    0   0 -Ws 570  67 -WP    0   0 -Wl "<< CONSOLE >>" -WL consol
e -C
clock      -Wp  568   0 -Ws 192  31 -WP  572   0 -Witextedit   -Wp receive 283
165 -Ws 584 576 -WP  708   0 -Wi
mailtool   -Wp  568  66 -Ws 584 555 -WP  776   0 -Wi
shelltool  -Wp  106  67 -Ws 638 790 -WP  640   0
shelltool  -Wp  126 164 -Ws 570 427 -WP   64   0
```

Deleting Words

Use your UNIX *word erase* character (usually CTRL-W) to delete the previous word.

Figure 8-17 *Deleting a Word*



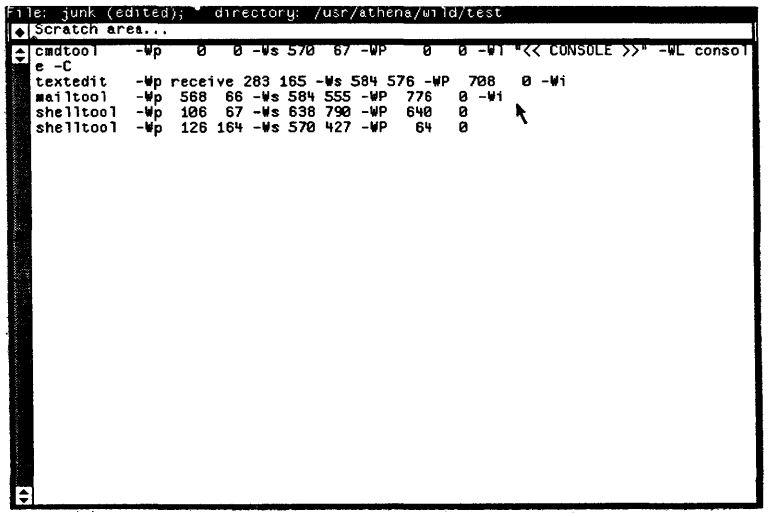
```
File: junk (edited);   directory: /usr/athena/wild/test
Scratch area...
cmdtool     -Wp    0    0 -Ws 570  67 -WP     0    0 -Wl "<< CONSOLE >>" -WL consol
e -C
clock       -Wp  568    0 -Ws 192  31 -WP   572    0 -textedit   -Wp receive 283 16
5 -Ws 584 576 -WP  708   0 -Wi
mailtool    -Wp  568   66 -Ws 584 555 -WP   776    0 -Wi
shelltool   -Wp  106   67 -Ws 638 790 -WP   640    0
shelltool   -Wp  126  164 -Ws 570 427 -WP    64    0
```

As with the rubout character, (SHIFT) toggles the direction of the delete, deleting the next word to the right of the caret, instead of to the left.

Deleting Lines

Use your *line kill* character (usually (CTRL-U)) to delete the previous line.

Figure 8-18 *Deleting a Line*



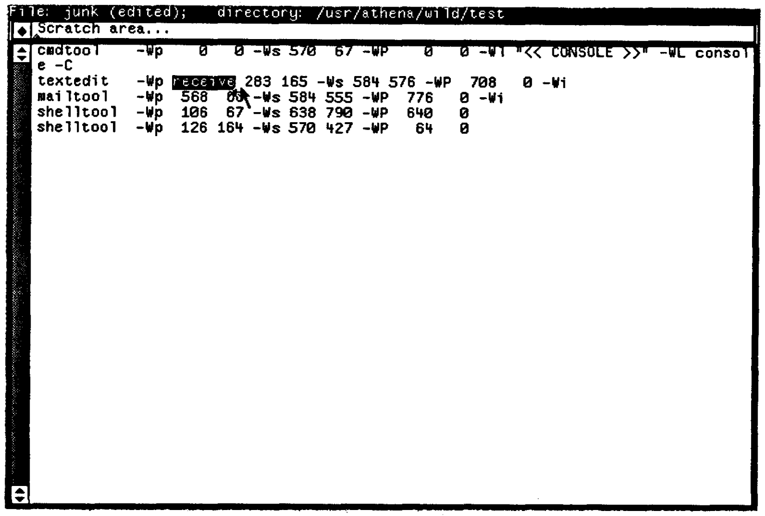
```
File: junk (edited);   directory: /usr/athena/wild/test
Scratch area...
cmdtool     -Wp    0    0 -Ws 570  67 -WP     0    0 -Wl "<< CONSOLE >>" -WL consol
e -C
textedit    -Wp receive 283 165 -Ws 584 576 -WP  708   0 -Wi
mailtool    -Wp  568   66 -Ws 584 555 -WP   776    0 -Wi
shelltool   -Wp  106   67 -Ws 638 790 -WP   640    0
shelltool   -Wp  126  164 -Ws 570 427 -WP    64    0
```

As with the rubout character, (SHIFT) toggles the direction of the delete, deleting the rest of the line, rather than the previous line.

Deleting with a Function Key

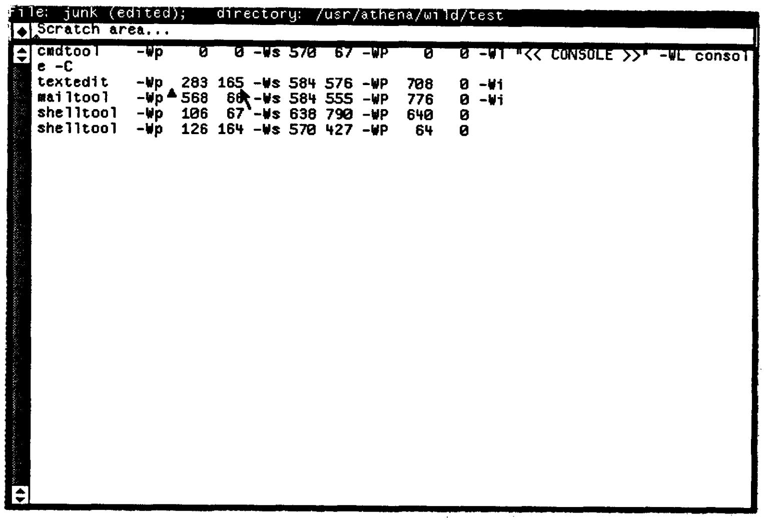
Another way to delete a character or characters is to select them, then press the [Delete] function key (usually [L10]).

Figure 8-19 *Selection of Text to Be Deleted*



```
File: junk (edited);    directory: /usr/athena/wild/test
Scratch area...
cmdtool     -Wp     0    0 -Ws 570  67 -WP     0    0 -Wl "<< CONSOLE >>" -WL consol
e -C
textedit    -Wp receive 283 165 -Ws 584 576 -WP  708    0 -Wi
mailtool    -Wp  568   68 -Ws 584 555 -WP  776    0 -Wi
shelltool   -Wp  106   67 -Ws 638 790 -WP  640    0
shelltool   -Wp  126 164 -Ws 570 427 -WP   64    0
```

Figure 8-20 *Deleting the Selection with a Function Key*



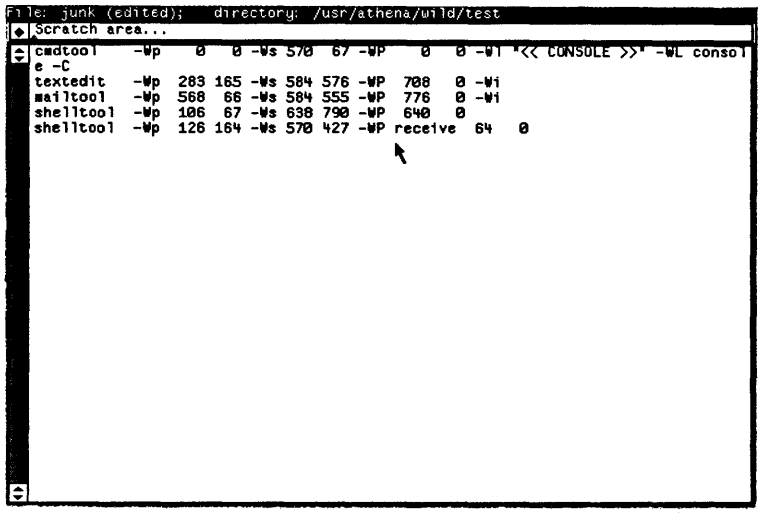
```
File: junk (edited);    directory: /usr/athena/wild/test
Scratch area...
cmdtool     -Wp     0    0 -Ws 570  67 -WP     0    0 -Wl "<< CONSOLE >>" -WL consol
e -C
textedit    -Wp  283 165 -Ws 584 576 -WP  708    0 -Wi
mailtool    -Wp  568   68 -Ws 584 555 -WP  776    0 -Wi
shelltool   -Wp  106   67 -Ws 638 790 -WP  640    0
shelltool   -Wp  126 164 -Ws 570 427 -WP   64    0
```

You can type [CTRL-D], instead of [Delete], if you find that easier.

Reinserting Text Deleted with a Function Key

You can use the [Get] function key (usually [L8]), or [CTRL-G], to *reinsert* the last selection that you deleted (from any text subwindow) at the caret.

Figure 8-21 *Reinserting the Deleted Selection*



```
File: junk (edited);    directory: /usr/athena/wild/test
Scratch area...
cmdtool    -Wp    0   0 -Ws 570  67 -WP    0   0 -Wi "<< CONSOLE >>" -WL consol
e -C
textedit   -Wp  283 165 -Ws 584 576 -WP  708   0 -Wi
mailtool   -Wp  568  66 -Ws 584 555 -WP  776   0 -Wi
shelltool  -Wp  106  67 -Ws 638 790 -WP  640   0
shelltool  -Wp  126 164 -Ws 570 427 -WP receive  64   0
```

Whenever you delete a selection, the text facility stores it on the *shelf*, like storing some jam on a shelf in your kitchen for later use.

[Get] copies the contents of the shelf to the insertion point.

Moving a Selection

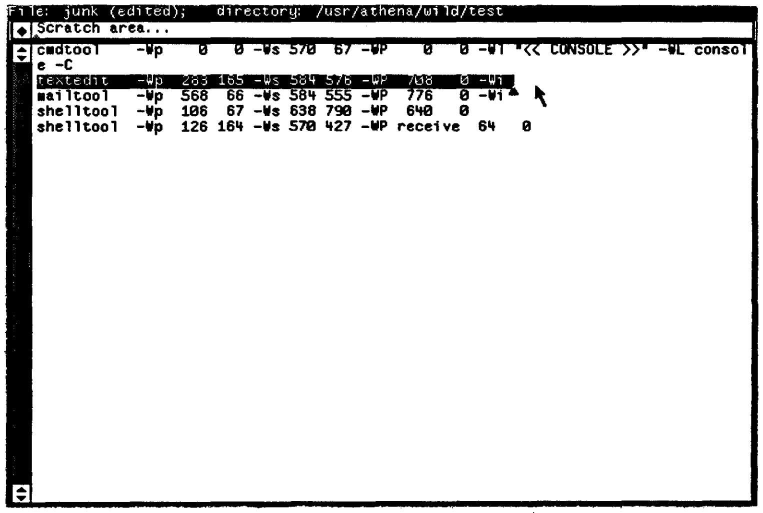
To *move* a selection from one location to another in the same text subwindow, or between different text subwindows:

- Make the selection.
- Press [Delete].
- Select the new insertion point.
- Press [Get].

Copying a Selection

The [Put] function key (usually [L6]) stores selected characters on the shelf without deleting them. Thus, you can *copy* characters from one place to another by selecting them.

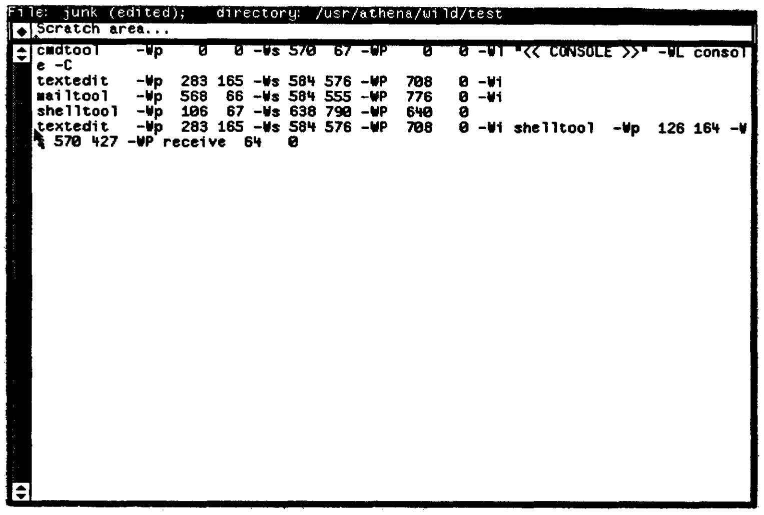
Figure 8-22 *Selecting Text to Be Copied*



```
File: junk (edited);    directory: /usr/athena/wild/test
Scratch area...
cmdtool    -Wp    0    0 -Ws 570  67 -WP    0    0 -Wl "<< CONSOLE >>" -WL consol
e -C
textedit   -Wp  283  165 -Ws 584 576 -WP  708    0 -Wi
mailtool   -Wp  568   66 -Ws 584 555 -WP  776    0 -Wi
shelltool  -Wp  106   67 -Ws 638 790 -WP  640    0
shelltool  -Wp  126  164 -Ws 570 427 -WP receive  64    0
```

Then press (Put). Select the new insertion point, and press (Get) to copy from the shelf to the insertion point on the screen.

Figure 8-23 *Copying the Text Selection*



```
File: junk (edited);    directory: /usr/athena/wild/test
Scratch area...
cmdtool    -Wp    0    0 -Ws 570  67 -WP    0    0 -Wl "<< CONSOLE >>" -WL consol
e -C
textedit   -Wp  283 165 -Ws 584 576 -WP  708    0 -Wi
mailtool   -Wp  568  66 -Ws 584 555 -WP  776    0 -Wi
shelltool  -Wp  106  67 -Ws 638 790 -WP  640    0
textedit   -Wp  283 165 -Ws 584 576 -WP  708    0 -Wi shelltool  -Wp  126 164 -W
 570 427 -WP receive  64    0
```

Replacing Text Selections

To *replace* a text selection with some other text:

- Select the text you wish to replace.
- Press (Delete).

- Insert the new characters, since the insertion point is in the place where the deleted characters were.

More About Move, Copy, and Replace

You can perform all variations of copy, replace, and move using the Delete, Put and Get.

You'll learn more efficient, but more complicated, ways to do moves, copies, replaces, and exchanges in the chapter on advanced editing (Chapter 9).

Compatibility Between `textedit` and `shelltool`

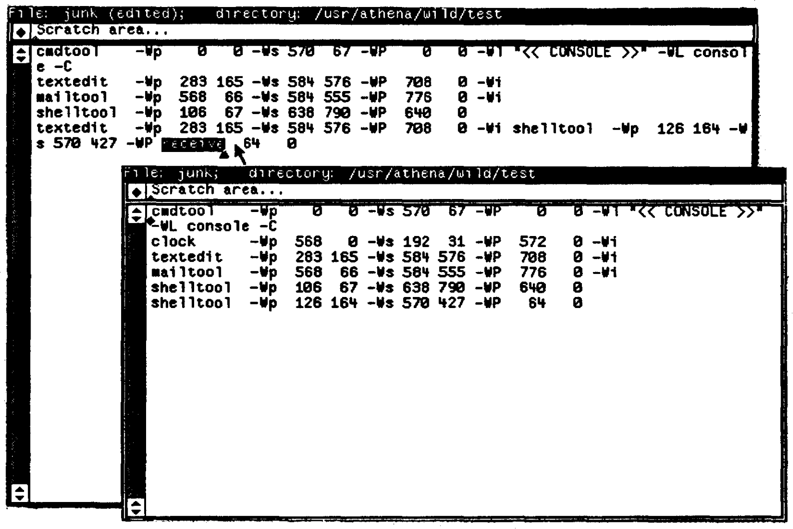
You can use Get and Put the same in `shelltools`, as in text subwindows (Delete and the other functions keys don't function in a `shelltool` since `shelltools` don't support editing).

Copying from One Window to Another

You can copy a selection from one window (of any type supported by the text facility, including `cmdtool` and `textedit`) to another window using the same mechanism you used to copy a selection from one spot to another within a single window:

- Make the selection in one window.
- Press Put.
- Select the insertion point in the other window.
- Press Get.

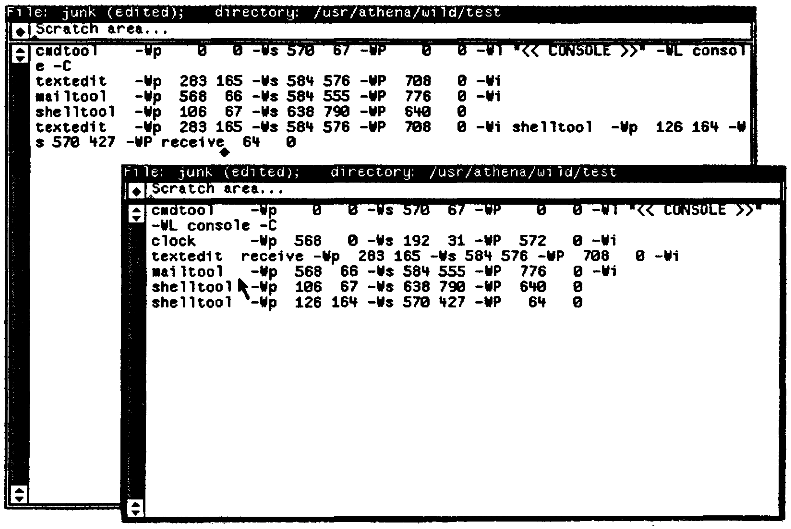
Figure 8-24 *Making a Selection in One Window*



```
File:  junk (edited);    directory: /usr/athena/wild/test
Scratch area...
cmdtool     -Wp    0    0 -Ws 570  67 -WP    0    0 -Wl "<< CONSOLE >>" -WL consol
e -C
textedit    -Wp  283 165 -Ws 584 576 -WP  708    0 -Wi
mailtool    -Wp  568  66 -Ws 584 555 -WP  776    0 -Wi
shelltool   -Wp  106  67 -Ws 638 790 -WP  640    0
textedit    -Wp  283 165 -Ws 584 576 -WP  708    0 -Wi shelltool  -Wp  126 164 -W
s 570 427 -WP receive 64   0
```

```
File:  junk;    directory: /usr/athena/wild/test
Scratch area...
cmdtool     -Wp    0    0 -Ws 570  67 -WP    0    0 -Wl "<< CONSOLE >>"
-WL console -C
clock       -Wp  568    0 -Ws 192  31 -WP  572    0 -Wi
textedit    -Wp  283 165 -Ws 584 576 -WP  708    0 -Wi
mailtool    -Wp  568  66 -Ws 584 555 -WP  776    0 -Wi
shelltool   -Wp  106  67 -Ws 638 790 -WP  640    0
shelltool   -Wp  126 164 -Ws 570 427 -WP   64    0
```

Figure 8-25 *Copying the Selection into the Other Window*



```
File:  junk (edited);    directory:  /usr/athena/wild/test
Scratch area...
cmdtool    -Wp    0    0 -Ws 570  67 -WP    0    0 -Wl "<< CONSOLE >>" -WL consol
e -C
textedit   -Wp  283 165 -Ws 584 576 -WP  708   0 -Wi
mailtool   -Wp  568  66 -Ws 584 555 -WP  776   0 -Wi
shelltool  -Wp  106  67 -Ws 638 790 -WP  640   0
textedit   -Wp  283 165 -Ws 584 576 -WP  708   0 -Wi shelltool  -Wp  126 164 -W
s 570 427 -WP receive  64   0
```

```
File:  junk (edited);    directory:  /usr/athena/wild/test
Scratch area...
cmdtool    -Wp    0    0 -Ws 570  67 -WP    0    0 -Wl "<< CONSOLE >>"
-WL console -C
clock      -Wp  568   0 -Ws 192  31 -WP  572   0 -Wi
textedit  receive -Wp  283 165 -Ws 584 576 -WP  708   0 -Wi
mailtool   -Wp  568  66 -Ws 584 555 -WP  776   0 -Wi
shelltool  -Wp  106  67 -Ws 638 790 -WP  640   0
shelltool  -Wp  126 164 -Ws 570 427 -WP   64   0
```

Undoing Editing Operations

Another useful operation is [Undo] (usually [L4]). When you press [Undo], you undo all of the edits you have made in the current text subwindow since the last time that you made a selection, or typed [Get], [Put], or [Delete].

Note: Each window keeps its own [Undo] information, unlike the shelf which all of the windows share. Currently, there is no way to [Undo] an [Undo].

Another [Undo] undoes the changes you made back through the time interval since the previous insertion point change. This [Undo] behavior continues back in time by changed insertion point intervals up to the limit specified when you created the text subwindow (the default value is 50 intervals).

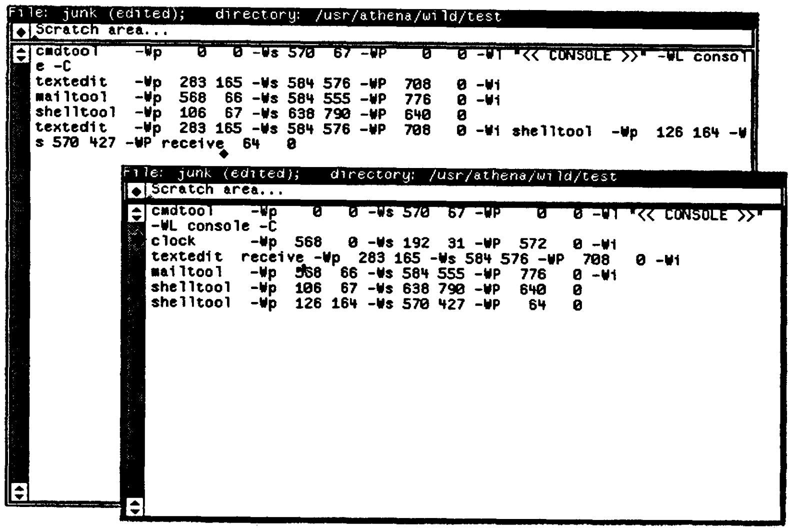
Scrolling Text

Note: This material about scrolling text subwindows applies to all subwindows that support scrolling (have scrollbars) that you'll encounter in this manual and in other SunView manuals.

Sometimes, the text you're editing is too large to be entirely visible within a text subwindow. In many cases, you can resize the window, but most often you probably want to *scroll* the text through the window. Think of the window as a porthole permitting you to access part of a larger mass of text. To access other parts of the text, you have to move the text relative to the window.

At the left of each text subwindow is a gray column called the *scrollbar*, which contains a darker gray area called the *bubble*. The bubble indicates the relative size and location of that portion of the document that is currently visible. For example, if the bubble is about one fourth the size of the scrollbar, the document is about four windows in length.

Figure 8-26 *The Scrollbar and Scroll Bubble*



```
File:  junk (edited);    directory:  /usr/athena/wild/test
Scratch area...
cmdtool     -Wp     0    0 -Ws 570   67 -WP     0    0 -Wl "<< CONSOLE >>" -WL consol
e -C
textedit    -Wp  283 165 -Ws 584 576 -WP  708    0 -Wi
mailtool    -Wp  568  66 -Ws 584 555 -WP  776    0 -Wi
shelltool   -Wp  106  67 -Ws 638 790 -WP  640    0
textedit    -Wp  283 165 -Ws 584 576 -WP  708    0 -Wi shelltool  -Wp  126 164 -W
s 570 427 -WP receive  64   0
```

```
File:  junk (edited);    directory:  /usr/athena/wild/test
Scratch area...
cmdtool     -Wp     0    0 -Ws 570   67 -WP     0    0 -Wl "<< CONSOLE >>"
-WL console -C
clock       -Wp  568   0 -Ws 192  31 -WP  572   0 -Wi
textedit  receive -Wp  283 165 -Ws 584 576 -WP  708   0 -Wi
mailtool    -Wp  568  66 -Ws 584 555 -WP  776   0 -Wi
shelltool   -Wp  106  67 -Ws 638 790 -WP  640   0
shelltool   -Wp  126 164 -Ws 570 427 -WP   64   0
```

At the top and bottom of the scrollbar, you can see a square that contains up/down arrows. These squares are *scroll buttons*.

Both scroll buttons operate the same way; the bottom button is a convenience when you don't want to move the cursor all the way to the top of the window when you're editing near the bottom.

Scrolling Forward — Text Moves Up

To scroll the display *forward* one line — text in window moves up — click [LEFT] in the scroll button.

Scrolling Backward — Text Moves Down

To scroll the display *backward* one line — contents of window moves down — click [RIGHT].

Scrolling a Page at a Time

You can scroll forward a *page* at a time, an amount equal to the height of the subwindow, by clicking [MIDDLE] in the scroll button, or scroll backward a page by clicking [MIDDLE] while holding down [SHIFT].

Some more advanced scrolling techniques appear in the chapter on advanced editing (Chapter 9).

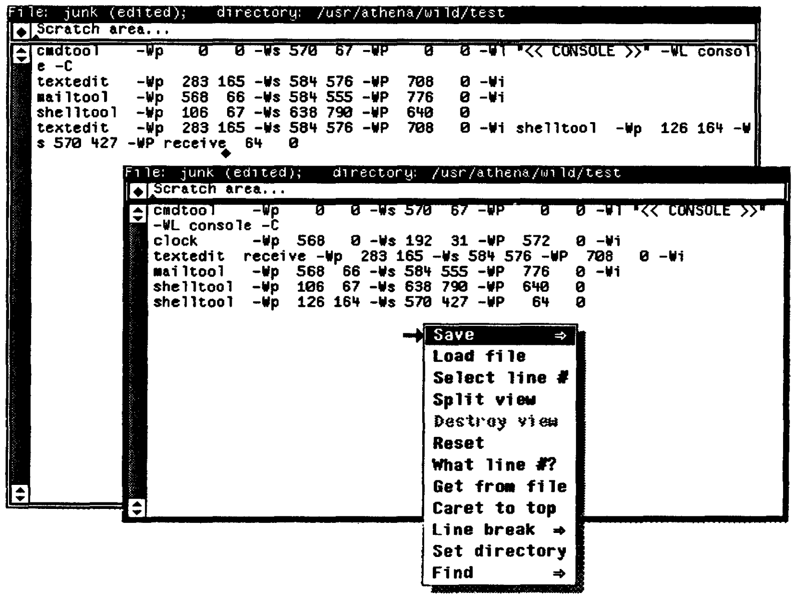
Saving Your Work

Save your work often when editing a file, because if the system crashes, you'll lose the work you've completed.

Saving Your Editing to the Current Filename

When you're ready to save your file editing work, pop up the text menu, and choose the `Save` menu item. This saves your work in the current filename associated with the text subwindow.

Figure 8-27 *Saving Your Editing Work to the Current Filename*



```
File:  junk (edited);    directory:  /usr/athena/wild/test
Scratch area...
cmdtool    -Wp    0   0 -Ws 570  67 -WP    0   0 -Wl "<< CONSOLE >>" -WL consol
e -C
textedit   -Wp  283 165 -Ws 584 576 -WP  708   0 -Wi
mailtool   -Wp  568  66 -Ws 584 555 -WP  776   0 -Wi
shelltool  -Wp  106  67 -Ws 638 790 -WP  640   0
textedit   -Wp  283 165 -Ws 584 576 -WP  708   0 -Wi shelltool  -Wp  126 164 -W
s 570 427 -WP receive  64   0
```

```
File:  junk (edited);    directory:  /usr/athena/wild/test
Scratch area...
cmdtool    -Wp    0   0 -Ws 570  67 -WP    0   0 -Wl "<< CONSOLE >>"
-WL console -C
clock      -Wp  568   0 -Ws 192  31 -WP  572   0 -Wi
textedit  receive -Wp  283 165 -Ws 584 576 -WP  708   0 -Wi
mailtool   -Wp  568  66 -Ws 584 555 -WP  776   0 -Wi
shelltool  -Wp  106  67 -Ws 638 790 -WP  640   0
shelltool  -Wp  126 164 -Ws 570 427 -WP   64   0
```

Note: When you're editing a file in a `textedit` text subwindow, the name of the file you are currently editing, if any, and the current working directory appear in the namestripe at the top of the window.

`textedit`'s Backup Files

`textedit` copies the original contents of the file, as of the last save or store, to a filename with the same name, except that it affixes a percent sign (`%`) to the end of it. `textedit` locates this backup file in the current directory at the time it copies the file.

These "`%`" files don't go away unless you delete them; but `textedit` overwrites them each time you save the file to the original filename. You can `Save` a file more than once and continue editing.

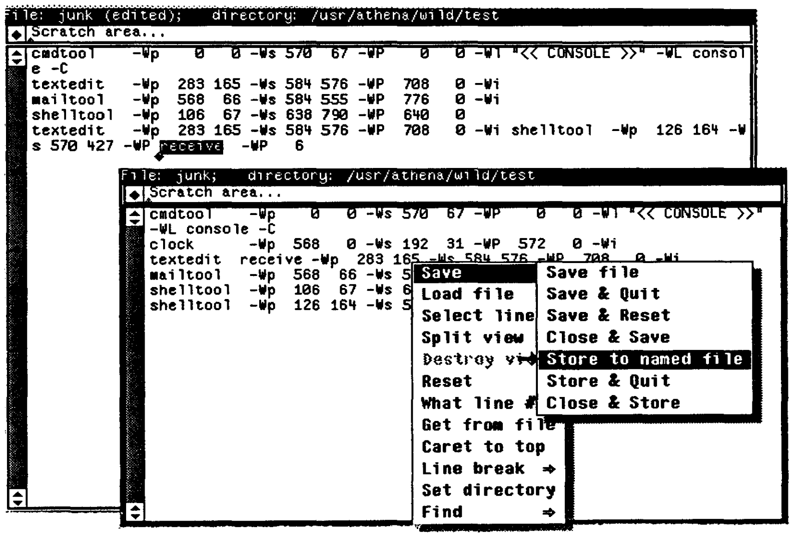
Storing Your Editing Work to Another Filename

To store your editing work to an existing or new file of your choice, use the `Store to named file` menu item. Follow these steps:

- Select the filename where you wish to save your editing work from any window on the screen. Specify the absolute or relative pathname of the file when you don't want to store to a file in the current directory.
- Pop up the text menu and hold down [RIGHT].
- Move the cursor into the `Save` menu item area.
- Move the cursor toward and through the arrow (=>) just to the right of the menu item on the walking menu to obtain a pull-right menu.

- Move the cursor to choose the `Store to named file` item in the pull-right menu.
- Release [RIGHT].

Figure 8-28 *Saving Your Editing Work to Another Filename*


```
File:  junk (edited);    directory: /usr/athena/wild/test
Scratch area...
cmdtool     -Wp    0    0 -Ws 570   67 -WP    0    0 -Wl "<< CONSOLE >>" -WL consol
e -C
textedit    -Wp  283 165 -Ws 584 576 -WP  708    0 -Wi
mailtool    -Wp  568  66 -Ws 584 555 -WP  776    0 -Wi
shelltool   -Wp  106  67 -Ws 638 790 -WP  640    0
textedit    -Wp  283 165 -Ws 584 576 -WP  708    0 -Wi shelltool  -Wp  126 164 -W
s 570 427 -WP receive  -WP   6
```

```
File:  junk;    directory: /usr/athena/wild/test
Scratch area...
cmdtool     -Wp    0    0 -Ws 570   67 -WP    0    0 -Wl "<< CONSOLE >>"
-WL console -C
clock       -Wp  568    0 -Ws 192  31 -WP  572    0 -Wi
textedit  receive -Wp  283 165 -Ws 584 576 -WP  708    0 -Wi
mailtool    -Wp  568   66 -Ws 5
shelltool   -Wp  106   67 -Ws 6
shelltool   -Wp  126  164 -Ws 5
```

A menu that has pull-right menus accessible through a right arrow (=>) is a walking menu, described in the earlier section on walking menus (section 8.2).

You've finished the section on basic editing with the text facility and you deserve a break if you want one.

8.4. Intermediate Editing

This section adds a few details to the basic editing that you learned in the last section.

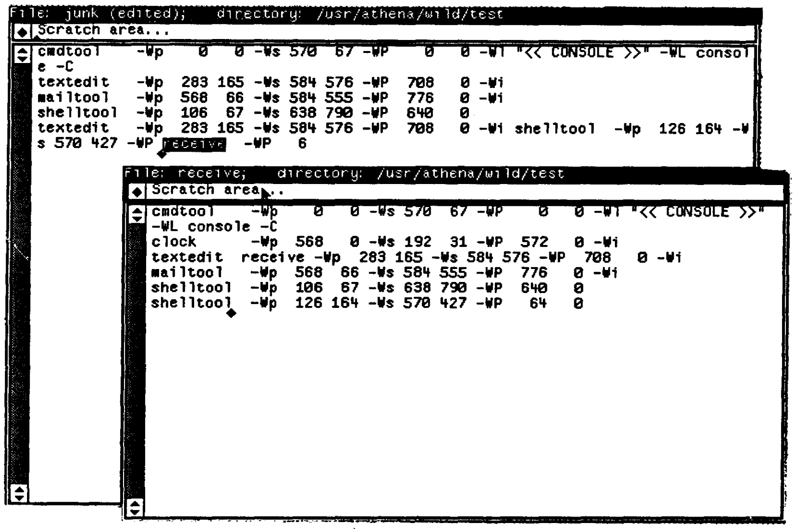
The Scratch Text Subwindow

The *scratch text subwindow*, or ''scratch area,'' is a text subwindow at the top of the `textedit` window in which you can type whatever you want without mixing it into the file you are editing.[23] So, when you want to select a filename, directory name, or whatever else:

- Move the cursor into the *scratch text subwindow*.
- Type the span of characters you want to select.
- Select the desired span of characters.

[23] You can specify the size of the scratch text subwindow using `defaultsedit`. For more information, see the section on `defaultsedit`, Section 11.1 .

Figure 8-29 *Moving the Cursor into the Scratch Text Subwindow*



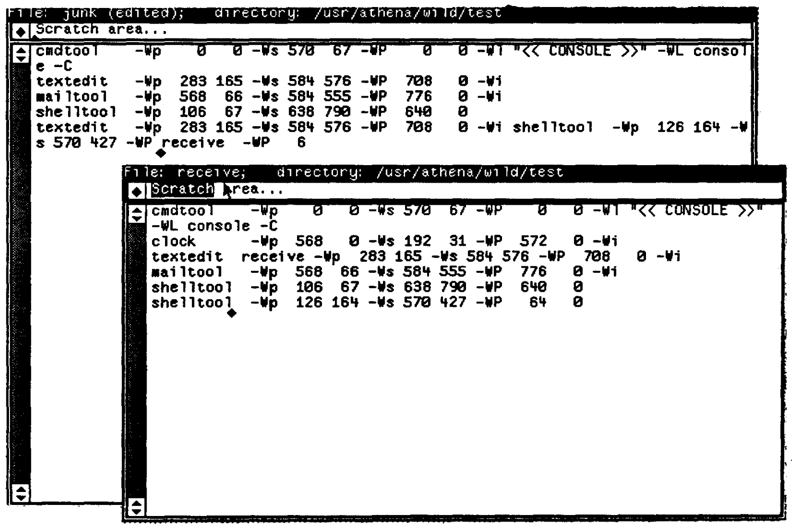
```
File: junk (edited);    directory: /usr/athena/wild/test
Scratch area...
cmdtool    -Wp    0   0 -Ws 570  67 -WP    0   0 -Wl "<< CONSOLE >>" -WL consol
e -C
textedit   -Wp  283 165 -Ws 584 576 -WP  708   0 -Wi
mailtool   -Wp  568  66 -Ws 584 555 -WP  776   0 -Wi
shelltool  -Wp  106  67 -Ws 638 790 -WP  640   0
textedit   -Wp  283 165 -Ws 584 576 -WP  708   0 -Wi shelltool  -Wp  126 164 -W
s 570 427 -WP receive  -WP   6
```

```
File: receive;    directory: /usr/athena/wild/test
Scratch area...
cmdtool     -Wp    0   0 -Ws 570  67 -WP    0   0 -Wl "<< CONSOLE >>"
-WL console -C
clock       -Wp  568   0 -Ws 192  31 -WP  572   0 -Wi
textedit  receive -Wp  283 165 -Ws 584 576 -WP  708   0 -Wi
mailtool    -Wp  568  66 -Ws 584 555 -WP  776   0 -Wi
shelltool   -Wp  106  67 -Ws 638 790 -WP  640   0
shelltool   -Wp  126 164 -Ws 570 427 -WP   64   0
```

Note: As an alternative, you can change the working directory of a `textedit` that has an empty primary subwindow in a way similar to the method for loading a file into an empty text subwindow described earlier — type the directory name in the text subwindow, followed by ESC.

You select the text just as you did in the primary subwindow, by clicking LEFT over one end of the span and MIDDLE to adjust the selection to the other end of the span.

Figure 8-30 *Selecting Within the Scratch Text Subwindow*



```
File: junk (edited);    directory: /usr/athena/wild/test
Scratch area...
cmdtool    -Wp    0   0 -Ws 570  67 -WP    0   0 -Wl "<< CONSOLE >>" -WL consol
e -C
textedit   -Wp  283 165 -Ws 584 576 -WP  708   0 -Wi
mailtool   -Wp  568  66 -Ws 584 555 -WP  776   0 -Wi
shelltool  -Wp  106  67 -Ws 638 790 -WP  640   0
textedit   -Wp  283 165 -Ws 584 576 -WP  708   0 -Wi shelltool  -Wp  126 164 -W
s 570 427 -WP receive  -WP   6
```

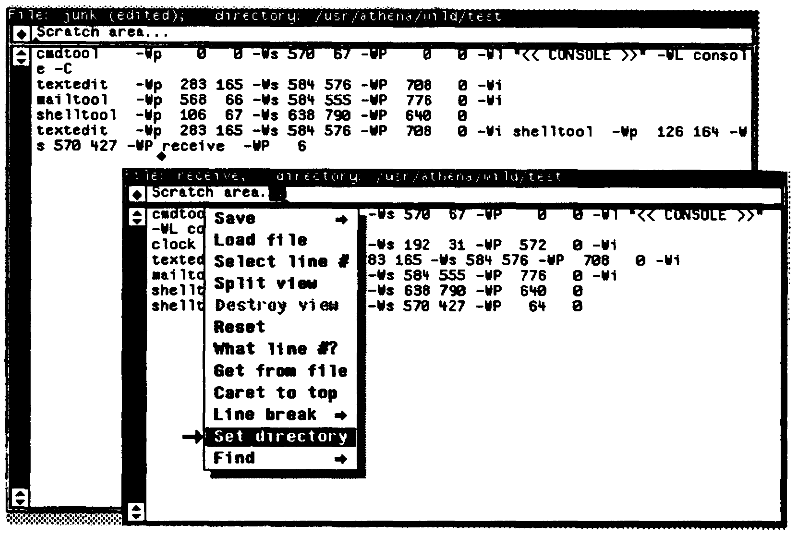
```
File: receive;    directory: /usr/athena/wild/test
Scratch area...
cmdtool     -Wp    0   0 -Ws 570  67 -WP    0   0 -Wl "<< CONSOLE >>"
-WL console -C
clock       -Wp  568   0 -Ws 192  31 -WP  572   0 -Wi
textedit  receive -Wp  283 165 -Ws 584 576 -WP  708   0 -Wi
mailtool    -Wp  568  66 -Ws 584 555 -WP  776   0 -Wi
shelltool   -Wp  106  67 -Ws 638 790 -WP  640   0
shelltool   -Wp  126 164 -Ws 570 427 -WP   64   0
```

Changing the Working Directory with the Text Menu

You can change the working directory of `textedit` at any time by choosing the `Set Directory` item on the text menu. The current working directory of the `textedit` appears in the namestripe, along with the name of the file you're editing. Basically you need to follow these steps:

- Select the directory name from any of the text subwindows or `shelltools` on your screen; if the directory name doesn't occur on the screen, you can type it in the scratch area and select it.
- Move the cursor into the text subwindow.
- Pop up the text menu by pressing and holding down [RIGHT].
- Choose the `Set Directory` menu item by moving the cursor over that item and releasing [RIGHT].

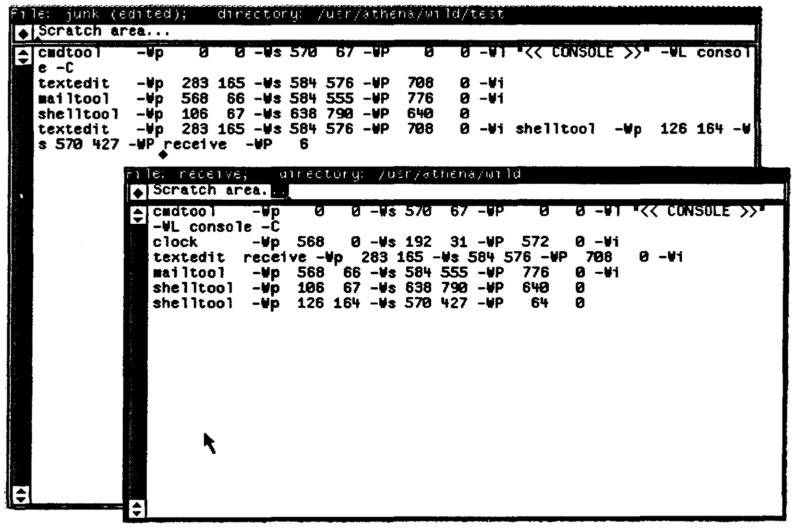
Figure 8-31 *Selecting the* `Set Directory` *Item on the Text Menu*


```
File:  junk (edited);    directory: /usr/athena/wild/test
Scratch area...
cmdtool     -Wp    0    0 -Ws 570  67 -WP    0    0 -Wi "<< CONSOLE >>" -WL consol
e -C
textedit    -Wp  283 165 -Ws 584 576 -WP  708   0 -Wi
mailtool    -Wp  568  66 -Ws 584 555 -WP  776   0 -Wi
shelltool   -Wp  106  67 -Ws 638 790 -WP  640   0
textedit    -Wp  283 165 -Ws 584 576 -WP  708   0 -Wi shelltool  -Wp  126 164 -W
s 570 427 -WP receive  -WP   6
```

```
File: receive,    directory: /usr/athena/wild/test
Scratch area.
cmdtoo -Ws 570  67 -WP    0    0 -Wi "<< CONSOLE >>"
-WL co
clock  -Ws 192  31 -WP  572   0 -Wi
texted 83 165 -Ws 584 576 -WP  708   0 -Wi
mailto -Ws 584 555 -WP  776   0 -Wi
shellt -Ws 638 790 -WP  640   0
shellt -Ws 570 427 -WP   64   0
```

The namestripe notifies you of the directory change from the directory `/usr/athena/wild/test` to the directory `/usr/athena/wild`.

Figure 8-32 `textedit` *Namestripe after Directory Change*



```
File: junk (edited);    directory: /usr/athena/wild/test
Scratch area...
cmdtool     -Wp    0    0 -Ws 570  67 -WP    0    0 -Wi "<< CONSOLE >>" -WL consol
e -C
textedit    -Wp  283 165 -Ws 584 576 -WP  708    0 -Wi
mailtool    -Wp  568  66 -Ws 584 555 -WP  776    0 -Wi
shelltool   -Wp  106  67 -Ws 638 790 -WP  640    0
textedit    -Wp  283 165 -Ws 584 576 -WP  708    0 -Wi shelltool  -Wp  126 164 -W
s 570 427 -WP receive  -WP   6
```

```
File: receive;    directory: /usr/athena/wild
Scratch area...
cmdtool     -Wp    0    0 -Ws 570  67 -WP    0    0 -Wi "<< CONSOLE >>"
-WL console -C
clock       -Wp  568    0 -Ws 192  31 -WP  572    0 -Wi
textedit  receive -Wp  283 165 -Ws 584 576 -WP  708    0 -Wi
mailtool    -Wp  568  66 -Ws 584 555 -WP  776    0 -Wi
shelltool   -Wp  106  67 -Ws 638 790 -WP  640    0
shelltool   -Wp  126 164 -Ws 570 427 -WP   64    0
```

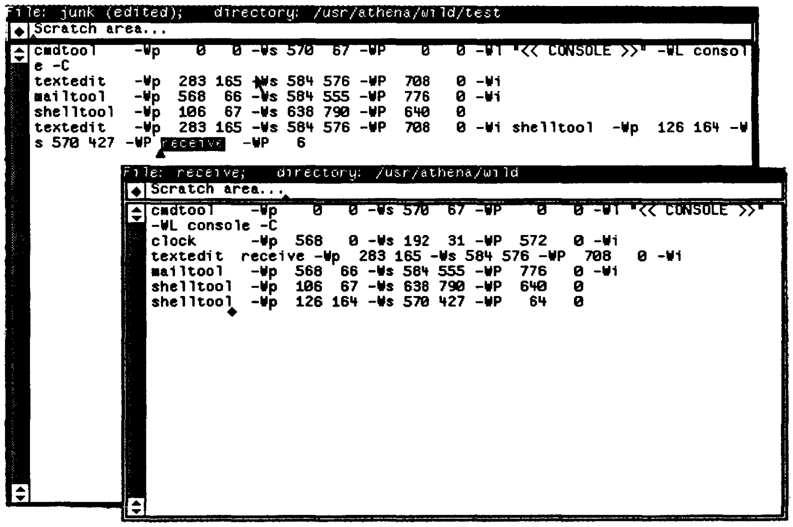
Loading a File with the Text Menu

To load a file with the text menu:

- Select the filename from any of the text subwindows or `shelltools` on your screen; if the filename doesn't occur on the screen, you can type it in the scratch text subwindow and select it.
- Move the cursor into the primary `textedit` subwindow.
- Pop up the text menu and choose the `Load file` item.

Here is the illustrated description — select a filename, for example `units`.

Figure 8-33 *Selecting a Filename*

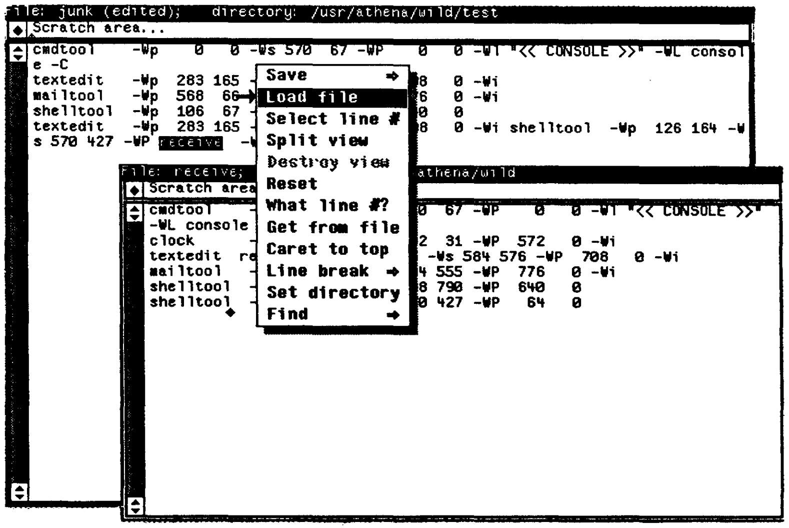


```
File: junk (edited);    directory: /usr/athena/wild/test
Scratch area...
cmdtool     -Wp     0    0 -Ws 570  67 -WP     0    0 -Wl "<< CONSOLE >>" -WL consol
e -C
textedit    -Wp   283 165 -Ws 584 576 -WP   708    0 -Wi
mailtool    -Wp   568  66 -Ws 584 555 -WP   776    0 -Wi
shelltool   -Wp   106  67 -Ws 638 790 -WP   640    0
textedit    -Wp   283 165 -Ws 584 576 -WP   708    0 -Wi shelltool  -Wp  126 164 -W
s 570 427 -WP receive   -WP    6
```

```
File: receive;    directory: /usr/athena/wild
Scratch area...
cmdtool     -Wp     0    0 -Ws 570  67 -WP     0    0 -Wl "<< CONSOLE >>"
-WL console -C
clock       -Wp   568    0 -Ws 192  31 -WP   572    0 -Wi
textedit  receive -Wp  283 165 -Ws 584 576 -WP  708    0 -Wi
mailtool    -Wp   568   66 -Ws 584 555 -WP   776    0 -Wi
shelltool   -Wp   106   67 -Ws 638 790 -WP   640    0
shelltool   -Wp   126  164 -Ws 570 427 -WP    64    0
```

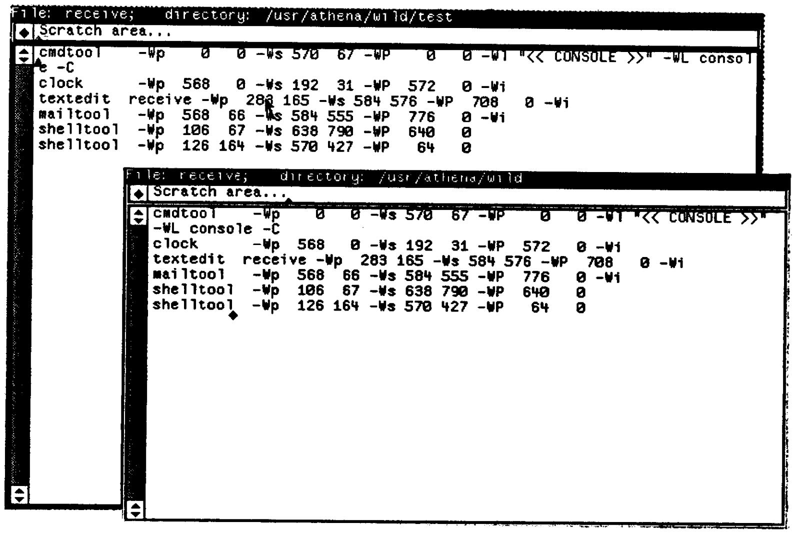
Pop up the text menu and choose the `Load file` item.

Figure 8-34 *Choosing the* `Load file` *Text Menu Item*



The `textedit` namestripe notifies you of the loaded filename, similar to the way it notifies you of directory changes.

Figure 8-35 `textedit` *Namestripe after Loading File*



```
File: receive;    directory: /usr/athena/wild/test
Scratch area...
cmdtool    -Wp    0    0 -Ws 570  67 -WP    0    0 -Wi "<< CONSOLE >>" -WL consol
e -C
clock      -Wp  568    0 -Ws 192  31 -WP  572    0 -Wi
textedit  receive -Wp  283 165 -Ws 584 576 -WP  708   0 -Wi
mailtool   -Wp  568   66 -Ws 584 555 -WP  776    0 -Wi
shelltool  -Wp  106   67 -Ws 638 790 -WP  640    0
shelltool  -Wp  126 164 -Ws 570 427 -WP   64    0
```

```
File: receive;    directory: /usr/athena/wild
Scratch area...
cmdtool    -Wp    0    0 -Ws 570  67 -WP    0    0 -Wi "<< CONSOLE >>"
-WL console -C
clock      -Wp  568    0 -Ws 192  31 -WP  572    0 -Wi
textedit  receive -Wp  283 165 -Ws 584 576 -WP  708   0 -Wi
mailtool   -Wp  568   66 -Ws 584 555 -WP  776    0 -Wi
shelltool  -Wp  106   67 -Ws 638 790 -WP  640    0
shelltool  -Wp  126 164 -Ws 570 427 -WP   64    0
```

Clearing the Text Subwindow with `reset`

You can clear the text subwindow by choosing the `Reset` text menu item.

This makes it possible to load a file by typing the filename, or to set the directory by typing the directory name in the text subwindow, followed by [ESC], as described above.

Creating a File from Scratch

If you want to create a new file from scratch, you can simply start inserting and editing text into an empty text subwindow. When you are finished, you can store the result on any file you like as described in the subsection on saving your work at the end of the section about basic editing (section 8.3). In other words, you don't have to start out by editing an existing file.

Leaving the Editor

Use the `Quit` menu item on the frame menu. The editor will not allow you to quit if you have made edits and not saved them. You can `Reset` when you want to quit without saving your work.

8.5. Summary of Basic and Intermediate Editing

Select	[LEFT] to select, [MIDDLE] to adjust.
Scroll	[LEFT] in scroll button to scroll forward one line, [RIGHT] to scroll backward, [MIDDLE] to scroll a page.
Insert	Type characters.
Delete	[Delete] function key or [CTRL-D]. With characters adjacent to the caret, you can use UNIX rubout, erase word, or line kill characters.

Reinsert deleted text	[Get] or [CTRL-G].
Replace	[Delete], type characters.
Move	[Delete], select destination, [Get].
Store on shelf	[Put].
Copy from shelf	[Get] or [CTRL-G].
Copy text from elsewhere	Select text, [Put], select destination, [Get].
Undo edits	[Undo] function key. Undoes all edits since last time insertion point changed.
Load a file	`Load file` item on text menu, or type name of file to empty text subwindow, followed by [ESC].
Save	`Save` or `Store to named file` item on text menu.
Exit	`Quit` item on frame menu.

9

Advanced Editing

9

Advanced Editing

The previous chapter presented beginning and intermediate text facility operations and features. This chapter presents more advanced features which allow you to perform these and other operations more efficiently and succinctly.

Learning more about the text menu will assist you in the advanced editing section that follows.

9.1. More About the Text Menu

The Text Menu contains the following items: `Save`, `Load file`, `Select line #`, `Split view`, `Destroy view`, `Reset`, `What line #?`, `Get from file`, `Caret to top`, `Line break`, `Set Directory`, `Find`, `Put then Get`, and `Edit On/Off`.

Some of these menu items are obscured in gray, meaning that you can't use them because they are inactive. For example, `Destroy view` only appears when there is more than one view — it doesn't make sense to destroy a view when there is only one view in the window.

Some menu items have => to their right, indicating that you can use them to pick hidden menu items on a pull-right menu (see the section on walking menus, Section 8.2).

Note: When using `textedit`, you can see the name of the current file and the current directory in the namestripe.

`Save =>` Save the contents of the text subwindow to the current filename. You can bring up a pull-right menu to pick these hidden items: `Save file`, `Save & Quit`, `Save & Reset`, `Close & Save`, `Store to named file`, `Store & Quit`, and `Close & Store`. `Save file` on the pull-right works the same way as the `Save` item on the original text menu. `Store to named file` allows you to store the contents of the text subwindow to a file other than the current one. Specify the name of file to which you want to store with the current selection, using either the absolute or relative pathname. If the indicated file already exists, the text facility asks you to confirm the operation. After storing, this file becomes the current file. `Save` writes to the new current file until you change it once again. The remaining four items are accelerators for performing two operations with a single action.

Note: As explained before, another way to load a file is to type into an empty text subwindow — namestripe displays `File: (NONE)` — the name of the file terminated by ESC.

`Load file`	Specify the name of the file you want to load with the current selection. If you've made any edits, the text facility asks you to confirm the operation.
`Select line #`	Treats current selection as a line number. Moves the caret to corresponding line and, if necessary, scrolls the text subwindow so that the line is visible. Useful to track down errors caught by a compiler.
`Split view`	Split a text subwindow into two subwindows. The text facility asks you to specify the boundary between the subwindows. You can scroll the two subwindows independently. However, they view the same document, and share a common primary and secondary selection and insertion point; the edits and selections you make in one subwindow are reflected in the other. You can split the subwindows again, and resize them the same way you resize any other subwindow. In other words, press CTRL-MIDDLE on the subwindow boundary and adjust the boundary to where you want it.
`Destroy view`	Destroy one of the views. Active only when more than one subwindow views the document.
`Reset`	Clear the text subwindow when you haven't made any edits. Otherwise, the text facility asks for confirmation, discarding all edits and returning to the state as of the last `Save` or `Store`, upon receipt of your confirmation.
`What line #?`	Display the line number of the (start of) the current selection. Useful in conjunction with `dbx`.
`Get from file`	Insert, at the caret, the contents of file named by the primary selection. If the primary selection is pending-delete, replace it with the contents of the file.
`Caret to top`	Reposition the text subwindow so that the line containing the caret is at the top of the window. Useful when the caret is not visible.
`Line break =>`	Normally the editor wraps long lines (see the section on wrapping long lines, Section 9.5). This menu item allows the user to switch from

	wrapping to clipping long lines or, when you pick from the pull-right menu, specifies `Clip lines` and `Wrap lines` explicitly. Only one of these pull-right items is active at a time; the other is obscured in gray.
`Set directory`	Set the working directory to the primary selection. Similar to the `cd` command typed to a shell whose current directory is the editor's current directory. The primary selection must specify an absolute or relative pathname.
`Find =>`	Search for the primary selection, or for the contents of the shelf when there is no primary selection. `Find` has four hidden items: `Find, forward`, the default, `Find, backward`, to search in reverse, `Find shelf, forward`, to search explicitly for the contents of the shelf, and `Find shelf, backward`, to search in reverse for the contents of the shelf.
`Put then Get`	Copies the current selection to the insertion point. When there is no current selection, `Put then` is obscured in gray, so as to perform only the `Get` operation, copying the contents of the shelf, if any, to the insertion point.
`Edit On/Off`	Permits (`Edit On`), or disallows (`Edit Off`), text facility modifications behind the read-only boundary of command subwindows. When you've chosen `Edit On`, you can't alter your command prompt or command history. Operations that don't modify the command prompt or command history still function with `Edit On`.

9.2. Advanced Editing

This section describes in detail:

- Advanced selecting
- Advanced scrolling
- Splitting views
- Advanced editing operations
- Searching for character spans with (Find)
- Repeating operations with (Again)
- Editing in tty subwindows (`shelltool`)
- Editing in command subwindows (`cmdtool`)
- Mousing ahead

- Wrapping long lines
- The necessity of saving your editing work
- Filters and extensibility
- Various text options

But first some new terminology.

Terminology

Here are some additional text selection terms you will need to know.

Multi-Clicking	Pressing a mouse key more than once in quick succession at the same location. You can set the maximum time between lifting up a key and pressing down the next key within a multi-clicking operation using `defaultsedit`; the default interval is 0.39 seconds. You can set the maximum distance of movement between clicks using `defaultsedit` as well; the default distance is 3 pixels.
Secondary Selection	A selection you make while holding down a function key. The selection, indicated by underlining the selecting characters when using [Put] or [Find], and by obscuring in gray when using [Delete], lasts only as long as you hold down the function key. When you release the function key, it operates on the temporarily selected characters, rather than on the primary selection.
Pending-Delete Selection	A selection, either primary or secondary, you make while holding down the [CTRL] key. You automatically delete pending-delete selections with a subsequent insertion or copy. Pending-delete selections are obscured in gray.

Note: A primary pending-delete selection is obscured by a darker gray than a secondary pending-delete selection.

Advanced Selecting

These selection techniques expand on the beginning selecting techniques described in the last chapter.

Level of Selection

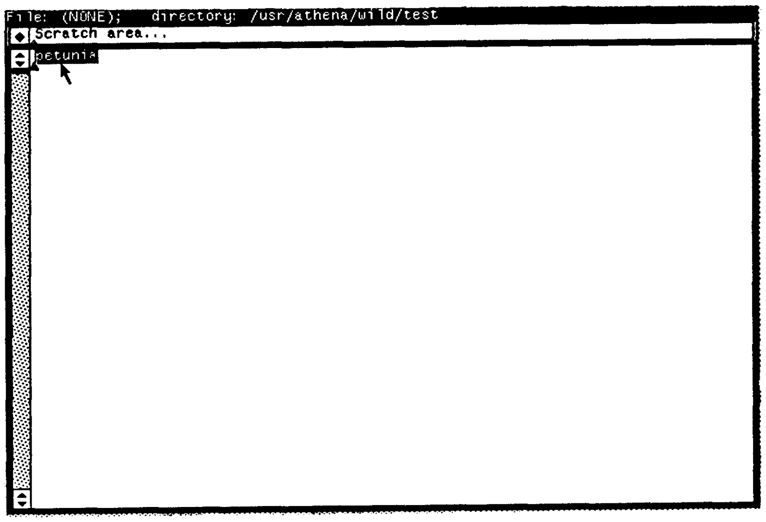
Each selection has a *level*, referring to the units that compose the selection.

Note: A *word* is a sequence of one or more alphanumeric characters beginning and ending in any non-alphanumeric character.

For example, you could select the previous sentence as a span of 79 characters, 13 words, or one line. By default, you make selections at the character level. However, you can specify higher levels of selection, for example words or lines, by multi-clicking. You can select a *word* by pointing at any character in that word and double-clicking [LEFT]. The main attraction of word selection is speed and convenience; you don't have to select carefully the first and last characters in the word. To select an entire line, point anywhere within the line and triple-click.

When you make a selection by multi-clicking, the caret appears at the end of the word closest to where the cursor was. For example, when you place the cursor anywhere to the left of the letter `u` in the word `petunia`, and double click, the caret will appear in front of the letter `p`.

Figure 9-1 *Example of Multi-Clicking To Select a Word*



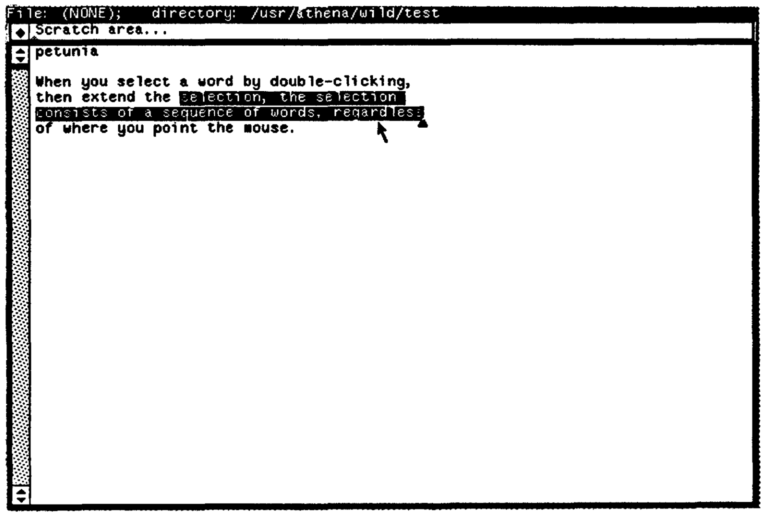
```
File: (NONE);   directory: /usr/athena/wild/test
Scratch area...
petunia
```

Using multi-clicking, you can easily position the caret before or after a word, or at the beginning of a line.

You can adjust a selection in units of its current level. When you select a word by double-clicking, then extend the selection, the selection consists of a sequence of words, regardless of where you position the mouse.

For example, to select in the previous sentence the span of characters that begins with the word `selection` and ends with the word `regardless`, point anywhere in the word `selection`, double-click (LEFT), then point *anywhere* in the word `regardless`, and click (MIDDLE).

Figure 9-2 *Adjusting a Selection at a Higher Level*



```
File: (NONE);    directory: /usr/athena/wild/test
Scratch area...
petunia

When you select a word by double-clicking,
then extend the selection, the selection
consists of a sequence of words, regardless
of where you point the mouse.
```

This method provides a very efficient method for selecting a span of words or lines.

Secondary Selections

A *secondary selection* is a selection you make while holding down a function key. The function key operates on the secondary selection when you release it.

Secondary selections, indicated by underlining, only last as long as you hold down a function key. During that time, you can adjust a secondary selection the same way you adjust a primary selection.

You can also scroll while making a secondary selection, as long as you continue to hold the function key down. That way, you can adjust the selection to encompass characters that aren't visible in the current subwindow.

Note: When you want to copy text from another window to the editing window, you must press the (Get) key in the editing window *before* you move the mouse from that window, otherwise you will shift attention to the other window and perform the (Get) operation there by mistake. In other words, the text facility performs the operation in the window where you first press the function key.

With secondary selections, you can specify operands for editing operations that require both an operand and a destination, such as copy or move. Secondary selections provide the ability to perform an operation without disturbing the current primary selection and insertion point.

You already learned one method for copying text from one place to another:

- Select the text
- Press (Put)
- Select the destination

- Press [Get]

A much easier way to accomplish the same operation is:

- Select the destination
- *Hold down* [Get] while selecting the desired text
- Release [Get]

If the caret is at the desired insertion point, then only the last two steps are necessary — simply hold down [Get] and reach out and select the text you want to copy.

You can also use secondary selections to express the operands for [Put] and [Delete], as well as for [Get].

For example, when you are inserting some text and you want to delete some text without disturbing the insertion point, simply hold the [Delete] key down and select the text you want to delete. When you release the [Delete] key, you delete the selected text, but the caret remains in its original position.

Note: If you ever notice that a text subwindows gets stuck so it makes only secondary selections, press the [Stop] function key to reset the subwindow.

Similarly, you can make a secondary selection while holding down the [Put] key, allowing you to specify where to insert the contents of the primary selection, instead of simply storing the primary selection on the shelf.

In conjunction with pending-delete selections, secondary selections provide abbreviated methods for copy and replace, move, and move and replace.

Pending-Delete Selections

When you want to replace a selection immediately after you make the selection, *pending-delete selections* make your job easier.

One frequent operation is to select some text, then type or copy its replacement. Pending-delete selections facilitate this more complex operation. In combination with secondary selections, they provide an extremely terse method for specifying a variety of common editing operations.

You can specify a pending-delete selection by holding down the [CTRL] key while selecting or adjusting a selection. When a selection is pending-delete, any insertion or transfer of characters deletes the contents of the selection.

Note: Remember that when you've used `setkeys` to put your function keys on the right side of the keyboard, you can type the [LINE FEED] key as an alternative [CTRL] key.

For example, to move text from one location to another:

- Place the caret at the desired location
- Hold down both the [Get] and [Ctrl] keys
- Select the text
- Release [Get]

When you release the [Get] key, you insert at the caret the text you specified by the secondary selection. In addition, you have deleted the pending-delete secondary selection from its original location.

You delete the contents of a pending-delete selection only when you insert characters subsequently; when you make a new selection without performing any operation or inserting any characters, you deselect the pending-delete selection without deleting it.

Pending-delete selections are indicated by a gray overlay. Think of the characters as being less visible because they are fading away, soon to be gone altogether.

You can change an existing selection to a pending-delete selection by holding the [CTRL] key down and adjusting the selection, in other words, clicking [MIDDLE] while pointing at one of its endpoints. The same character span becomes a pending-delete selection.

You can remove the pending-delete status of a selection by performing the reverse operation: adjust the selection by pointing at one of its endpoints while the [CTRL] key is *not* held down.

Aborting an Operation

When you start an operation, for example, hold down [Get] and select some text, then decide you don't want to complete the operation, press the [Stop] key (usually [L1]) to abort the operation.

[Stop] also aborts SunView operations like the frame menu items `Move` and `Resize`, as well as the mouse button accelerators for these operations.

As mentioned above, if you get stuck in a mode where all your selections appear as secondary selections, type [Stop] to reset the selection service.

Advanced Scrolling

You've learned how to scroll by lines or pages. By clicking the mouse buttons on the portion of the scroll bar that contains the scroll bubble, rather than the portion with the scroll buttons, you can scroll in arbitrary fractions of a page. The mouse buttons are the same:

[LEFT]
  Scroll forward (display moves up)

[RIGHT]
  Scroll backward (display moves down)

[MIDDLE]
  Scroll in larger units

However, the actual amount of scrolling depends on the position of the cursor in the scrollbar, as follows:

[LEFT]
  With the cursor in the scrollbar: move the line opposite the cursor to the top of the window

[RIGHT]
  With the cursor in the scrollbar: move the line at the top of the window to the position of the cursor

When you hold down [SHIFT], you scroll in the same direction as before, but with respect to the *bottom* of the window. This is especially useful when the upper portion of a window is covered by another window.

SHIFT-LEFT
: *With the cursor in the scrollbar*: move the line that at the bottom of the window to be opposite the cursor

SHIFT-RIGHT
: *With the cursor in the scrollbar*: move the line opposite the cursor to the bottom of the window

Use MIDDLE for *thumbing*, by analogy with what one does with a thick book such as a dictionary or telephone book.

For example, when you click MIDDLE while the cursor is one third of the way down from the top of the window, you scroll so that the text subwindow displays the portion of text one third of the way through the document.

Moving the cursor to the top of the scroll bar (but not into the scroll button) and clicking MIDDLE is a quick way to get to the beginning of the file. Similarly, moving the cursor to the bottom of the scroll bar and clicking MIDDLE positions the window at the end of the file.

MIDDLE
: *With the cursor in the scrollbar*: Scroll the window proportional to position of cursor in scrollbar.

Sometimes you may scroll to another part of a file by thumbing, and then want to return to the previous location. Just move the cursor into the scrollbar and click SHIFT-MIDDLE.

SHIFT-MIDDLE
: *With the cursor in the scrollbar*: Return the window to the location before the last "big jump," in other words, before the last thumb operation, before the last search that caused a scroll, or before the last SHIFT-MIDDLE of a scroll.

Before each of these operations, the text facility stores the current position in the document, so you can return to this position with SHIFT-MIDDLE.

Splitting

You can look at several different, possibly non-contiguous portions of the document you're editing by *splitting the view*, or dividing the text subwindow into multiple subwindows.

Choose the `Split this view` menu button on the text menu to split a text subwindow.

Each view operates as an independent text subwindow with respect to scrolling, but all views share a single caret, primary, and secondary selection. When you edit one subwindow, the edits you make are reflected in all of the subwindows, even if you can't see the relevant portion of the text in a particular subwindow.

You can get rid of any view, except the original one, by picking the `Destroy this view` menu button.

Adjust the size of the subwindows using the same method you use to move and resize the application:

MIDDLE
: *When cursor is over subwindow boundary*: move the subwindow

CTRL-MIDDLE
: *When cursor is over subwindow boundary*: resize the subwindow

Advanced Editing Operations

These advanced editing operations permit complex manipulation of text within multiple text subwindows.

Move, Copy, and Replace Using Secondary and Pending-Delete Selections

The use of secondary and pending-delete selections provides a streamlined way of moving, copying, and replacing text.

There are eight possible move, copy, and replace operations that depend on whether you're copying or moving, inserting or replacing, transferring from the current insertion point to somewhere else, or from somewhere else to the current insertion point.

All eight operations follow the same pattern:

- Use pending-delete selections for moving and replacing (hold down the CTRL key while selecting)
- Use standard selections for copying and inserting
- Use Get when going there-to-here, in other words, from somewhere else to the caret
- Use Put when going here-to-there, in other words, from the caret to somewhere else

Note: You may remember that a *tty subwindow* is a subwindow that includes a command interpreter, but lacks some text facility operations, like `shelltool`.

The copying method described below works for both text and tty subwindows. Move operations work when the destination is a text or ttysubwindow, but the source must be a text subwindow (because tty subwindows do not support the delete text operation). Conversely, for replace operations, the source can be either a text or tty subwindow, but the destination must be a text subwindow.

There-To-Here

There-to-here editing operations are the most frequent of the eight complex editing operations.

Note: Text items on a *panel*, or a control subwindow with buttons like in `mailtool`, also support a subset of these editing operations.

Copy From There to Here

To copy text to the caret:

- Hold down Get
- Select the text
- Release Get

To copy the contents of the shelf to the caret:

- Press and release [Get]

Move From There to Here

To move text to the caret:

- Hold down [CTRL] and [Get]
- Select the text
- Release [CTRL] and [Get]

Copy There, Replace Here

To copy text to replace the primary selection:

- Make the primary selection be pending-delete, either by holding down [CTRL] while selecting it in the first place, or by holding down [CTRL] and adjusting the selection
- Hold down [Get]
- Select the text to use as a replacement
- Release [Get]

Similarly, to replace the primary selection with the contents of the shelf:

- Make the primary selection pending-delete
- Press and release [Get]

Move From There, Replace Here

To move characters to replace the primary selection:

- Make the primary selection pending-delete
- Hold down [CTRL] and [Get]
- Select the text to use as a replacement
- Release [CTRL] and [Get]

Here-To-There

The here-to-there operations are basically the same as there-to-here, except that [Put] is used in place of [Get]:

Copy Here to There

To copy the primary selection to another location:

- Make a primary selection
- Hold down [Put]
- Select the location
- Release [Put]

To copy the primary selection to the shelf:

- Make a primary selection

- Press and release [Put]

Move Here to There

To move the primary selection to another location:

- Make the primary selection pending-delete
- Hold down [Put]
- Select the location
- Release [Put]

To move the primary selection to the shelf:

- Make the primary selection pending-delete
- Press and release [Delete]

Copy Here, Replace There

To copy the primary selection to replace characters at another location:

- Make a primary selection.
- Hold down [Put]
- Select the characters to replace while holding down [CTRL]
- Release [CTRL] and [Put]

Move Here, Replace There

To move the primary selection to replace characters at another location:

- Make the primary selection pending-delete
- Hold down [CTRL] and [Put]
- Select the characters to replace
- Release [CTRL] and [Put]

In each of the four here-to-there cases above, the caret stays where it is.

Sometimes you may want to copy or move the primary selection somewhere else and have the caret move to the new location as well. You can do this by storing the primary selection on the shelf, either by pressing [Put] or [Delete], moving the caret to the new location, and using [Get]:

Copy Here to There, Go There

To copy the primary selection to another location and have the caret go along with it:

- Make the primary selection
- Press and release (‘‘click’’) [Put]
- Select the destination insertion point
- Click [Get]

Move Here to There, Go There

To move the primary selection to another location and move the caret with it:

- Make the primary selection
- Click [Delete]
- Select the destination insertion point
- Click [Get]

Exchange

Another useful operation is *exchange*:

Exchange Here and There

To exchange the primary selection with some other text:

- Make the primary selection pending-delete
- Hold down [CTRL] and [Put]
- Select the other text
- Release [CTRL] and [Put]

You have completed a **Move Here, Replace There** as described above. The deleted text is now on the shelf. The caret is where the primary selection used to be.

- Click [Get] to insert the text you replaced from where you stored it on the shelf, thereby completing the exchange.

Searching for Text Selections

The text facility provides a [Find] operation, usually invoked by pressing the function key [L9]. You can also use the `Find` text menu item. The primary selection is the target of the search, unless you make a secondary selection while holding down the [Find] key.

When you haven't made a primary or secondary selection, [Find] searches for the contents of the shelf.

The search starts at the caret, skipping over the current selection, in the forward direction. When you want to search backwards, hold down the [SHIFT] key when you press [Find]. [Find] highlights the first match as the current primary selection. The caret moves to the end of the selection.

A match occurs when the exact same characters are found; there is no regular expression matcher yet.

When you reach the end of the file (or the beginning of the file, if the search is backwards), the search wrap arounds to the other end of the file and continues until it reaches its starting point. The caret stops blinking while the search is in progress. If no match is found, the caret will resume blinking at its original location.

The Again Operation

The text facility provides an (Again) operation, usually invoked by pressing the function key (L2). (Again) reexecutes all of the operations that you performed in a particular text subwindow since the last time you changed the insertion point.

Note: Scrolling the display has no effect on (Again).

For example, choose an operation you want to perform at several locations in a document, let's say, insert a string you've typed in the text subwindow or stored on the shelf. Perform the first insertion in the usual way. Perform subsequent insertions by positioning the caret where you want to insert the string and pressing (Again).

To perform the same bracketing operation in several places, let's say, insert a quotation mark (") before and after each of several items, one method is to:

- Position the caret in front of the first item
- Type a quotation mark
- Position the caret in front of the next item
- Press (Again)

And so on through the list. Then:

- Position the caret after the first item
- Type a quotation mark
- Position the caret after the next item
- Press (Again)

And so on...

To accomplish this more efficiently, use the shelf to avoid changing the insertion point. This enables you to use (Again) to repeat the combined operation of inserting material before and after the selected material. Here is how to do this:

- Select the first item
- (Delete) to delete the item and store it on the shelf
- Type quotation mark
- (Get) to retrieve what you deleted
- Type quotation mark
- Select the second item
- Press (Again) to repeat the previous four operations
- Select the third item
- Press (Again)

And so on..[24]

[24] For frequently used brackets, you may want to bind a function key to the invocation of an appropriate filter, as described below in the section on filters and extensibility (section 9.7).

[Again] ignores [Undo]. Thus, one very convenient use of [Again] is when you make an insertion at the wrong place. You simply [Undo], select, then press [Again].

Note: The information retained to enable [Again] is local to each text subwindow; edits performed in one text subwindow have no effect on the operation of [Again] in another text subwindow.

You can also use [Again] in conjunction with [Find] to iterate through a document, repeating a sequence of operations at several locations. (The [Find] command is counted as one of the operations that [Again] repeats, even though it changes the insertion point, because *you* did not change the insertion point.)

For example, you can search for a particular string, then perform some operation at that point, such as bracketing it with quotation marks using the technique described above. Then press [Again] to repeat both the search and the bracketing. (If the search fails, the rest of the operations that [Again] is repeating will not be executed.)

Editing Tty Subwindows and Panel Text Items

As described previously, tty subwindows support a subset of the editing operations, namely those that do not involve deletions. Secondary selections and storing the primary selection on the global shelf work the same as they do in text subwindows.

Thus, you can copy characters to and from text and tty subwindows using [Get] and [Put] in the manner described above.

Panel text items also support secondary selections and the use of [Get], [Put] and [Delete] in the manner described above, subject to two restrictions:

- You can only position the caret at the end of the item
- You can only select the entire item

Therefore, you can append text from a text subwindow, tty subwindow, or another panel text item to a panel text item that has a caret by holding down [Get] and selecting the text you want there.

Clicking [Get] copies the contents of the shelf to the end of the panel text item.

Similarly, you can copy the contents of a panel text item to another panel text item, or to a text or tty subwindow, or to the shelf, by selecting the item and using [Put] as described above.

Finally, you can type [Delete] to delete the contents of a panel text item and store it onto the shelf.

Editing Command Subwindows

A *command subwindow* is a text subwindow that permits a dialogue with the user, for example typing commands to `dbxtool`, or to a UNIX command line in `cmdtool`, which is a `shelltool` with text facility capabilities.

See the next chapter on the command facility, Chapter 10 , for more information.

9.3. Shift Toggles Direction of Function

Throughout the text facility user interface, we use SHIFT key to mean apply the inverse or negation of a function, or to apply the function in the inverse direction. There are currently several examples of this principle (and we expect to add more): SHIFT-Find means search in the backwards direction. SHIFT scroll means scroll with respect to the bottom of the window, rather than the top. SHIFT-DEL (rubout), SHIFT-CTRL-W (word erase), SHIFT-CTRL-U (line kill) means delete the next character, word, or rest of line.

9.4. Mousing Ahead

You don't have to wait for an operation to be completed before you specify the next one; you can *mouse ahead.* As you become more adept and confident with the text facility, you will probably start to mouse ahead more and more.

However, since the text facility interprets a mouse click at the time the editor processes the click, rather than when you make the click, be aware incomplete operations may affect the display in the area where you want to select.

For example, when you select a word in one line, Delete, then select a word in the next line and Delete before the editor has finished updating the display as a result of the first Delete, no problem.

However, when you click the mouse over another word that appears later in the *same* line before the display has been updated, then you might not be selecting the word you intended — the first Delete causes everything in that line to slide to the left.

So, if you want to perform a sequence of operations without waiting for the editor to complete each one, and these operations will affect the state of the display in such a way as to interfere with mousing ahead, perform the operations bottom to top and right to left. In this way, they will not interact with each other destructively.

9.5. Wrapping Long Lines

The default behavior for the text facility is to *wrap* long lines, in other words, the portion of a line that extends past the right edge of the window appears on the next line, starting at the left.

When you don't want wrapping of long lines, pick the `Line break` menu item on the text menu, or the `Clip lines` menu item on its pull-right menu. Then, the text subwindow *clips*, in other words, it doesn't display the characters that do not fit on the line.

To return to wrapping, pick the `Wrap lines` menu item on the pull-right menu.

Deletions in a wrapped line, particularly in the vicinity of where the wrapping occurs, can produce display anomalies. Tabs in long lines are particularly troublesome. If the display of a line gets into a weird state, you can redisplay the window by picking the `Redisplay` frame menu item.[25]

[25] You can eliminate this problem altogether by breaking long lines into shorter ones. Pipe the text through `fmt` with a filter, as described in the section on filters and extensibility (Section 9.7).

9.6. Remember to Save Your Work

When you have been editing for a long time, the editor's internal data structures may become sufficiently fragmented that you will notice a degradation in performance. Save your work, so performance will return to normal.

Frequent saving is probably a good idea in any case since, unlike for *vi*, if your machine crashes, the edits you made since the last store or save will be lost.

9.7. Filters and Extensibility

The text facility provides the ability to pipe the primary selection to an arbitrary program, or filter (as *stdin*), and insert the resulting output (*stdout*) at the caret. If the primary selection is in pending-delete, the output from the filter will replace it.

For example, piping a selection through `fmt` will cause all of the long lines to be truncated to 72 characters. Piping a selection through `indent` will format it according to the defaults specified in `~/.indent.pro` (see the `indent` Man Page, online or in the *Commands Reference Manual*).

You can bind a filter to an unused function key, in other words, either [R1] through [R14], [F1] through [F9], or if you have mapped the standard editing operations to the function keys on the right side of the keyboard, [L1] through [L10].

To find out how to specify this binding within a file called `~/.textswrc`, see the chapter on modifying subwindow behavior, Chapter 16 .

9.8. Text Options

You can specify a number of options to the text facility using `defaultsedit`.[26] The most frequently used options are:

`Auto_indent`	With `Auto_indent TRUE`, whenever you type a newline, the white space that appears at the beginning of the previous line is automatically inserted after the newline. This only affects newlines that you *type*, not the newlines you insert using the [Get] or [Put] function keys. The default setting for `Auto_indent` is `FALSE`.
`Adjust_is_pending_delete`	With `Adjust_is_pending_delete TRUE`, whenever you adjust the primary selection, it automatically becomes pending-delete, in other words, the text facility acts as though you were holding down [CTRL] key. The rationale behind this is that most of the time when you adjust a primary selection, you want to replace or delete it. The default setting for `Adjust_is_pending_delete` is `FALSE`.

[26] Read the section on `defaultsedit`, Section 11.1 .

`Scratch_window` *integer*	*Integer* is an integer parameter that specifies the height, in lines, of a scratch text subwindow for `textedit`. For example, when `Scratch_window=2`, then whenever you start `textedit`, it creates a two-line scratch text subwindow at the top, and a large text subwindow below it for editing. The default value is `1`, a one-line scratch window.
`Edit_back_char`	Your delete character. Default is (Del).
`Edit_back_word`	Your erase word character. Default is (CTRL-W).
`Edit_back_line`	Your line kill character. Default is (CTRL-U).
`Lower_context`	Minimum number of lines to maintain between caret and bottom margin of a text subwindow. Default is `2`.

9.9. Summary of Text Facility

Select	Double (LEFT) selects a word; Triple (LEFT) selects a line; (MIDDLE) adjusts a selection; (CTRL) while selecting or adjusting for pending-delete.
Scroll	(LEFT) in scroll bar to move corresponding line to top of window; (RIGHT) in scroll bar to move line at top of window to cursor; (SHIFT-LEFT) in scroll bar to move line at bottom of window to cursor; (SHIFT-RIGHT) in scroll bar to move corresponding line to bottom of window; (MIDDLE) in scroll bar to thumb; (SHIFT-MIDDLE) with cursor in scroll bar to return to previous location.
Multi-Clicking	To press a mouse key more than once in quick succession at the same location — see *Select*.
Copy to caret	Hold down (Get), make selection.
Move to caret	Hold down (Get), (CTRL) and make selection.
Replace at caret	(CTRL) and select primary, then type or (Get) selection.
Move and replace at caret	(CTRL) and select primary, hold down (Get), (CTRL) and make selection.
Copy primary elsewhere	Hold down (Put), select destination.
Move primary elsewhere	(CTRL) and select primary, hold down (Put), select destination.

Copy or move primary, replace elsewhere	Select or CTRL and select primary, hold down Put, CTRL and select destination.
Bracket a selection	Delete, insert leading text, Get, insert trailing text.
Get (Usually L8)	If you make secondary selection while key is down, copy selection to caret, otherwise copy shelf to caret. If secondary selection is pending-delete, delete it. If primary selection is pending-delete, delete it and copy to shelf.
Put (Usually L6)	If you make selection while key is down, copy primary selection to indicated place, otherwise copy primary selection to shelf. If secondary selection is pending-delete, delete it. If primary selection is pending-delete, delete it and copy to shelf.
Delete (Usually L10)	If you make selection while key is down, delete selected material, otherwise delete primary selection. Deleted material is copied to shelf.
Find (Usually L9)	If you make selection while key is down, search for selected material. Otherwise, search for primary selection. When no primary selection, search for contents of shelf. Search backwards when SHIFT is down, otherwise search forwards. Wrap around end, or beginning, of document.
Again (Usually L2)	Reexecute all edits performed since last time you moved the caret.
Undo (Usually L4)	Undo all edits since last time you selected, typed Get, Put, or Delete, or performed any operation except Again that uses Get, Put, or Delete. Another Undo undoes those edits made between that point and the previous Undo.

That completes all of the basic, intermediate, and advanced documentation on editing within the text facility.

10

The Command Facility

10 The Command Facility

The *command facility* is an extension of the text facility to provide a command interpreter along with the text facility functions. `cmdtool` is a command facility program that combines text editing features similar to `textedit` with a UNIX command interpreter similar to `shelltool`.

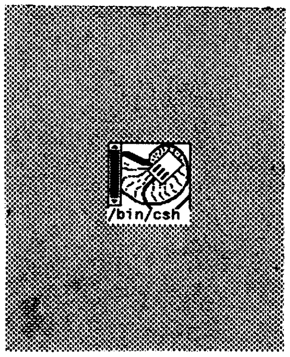
`dbxtool` also includes a *command subwindow*, or subwindow based on the command facility. Most of the information in the next section on `cmdtool` applies to all command subwindows.

10.1. `cmdtool`

Note: The `cmdtool` icon below contains a nautilus shell, referring to the *shell*, or command interpreter, and a scrollbar on the side, referring to its text facility capabilities.

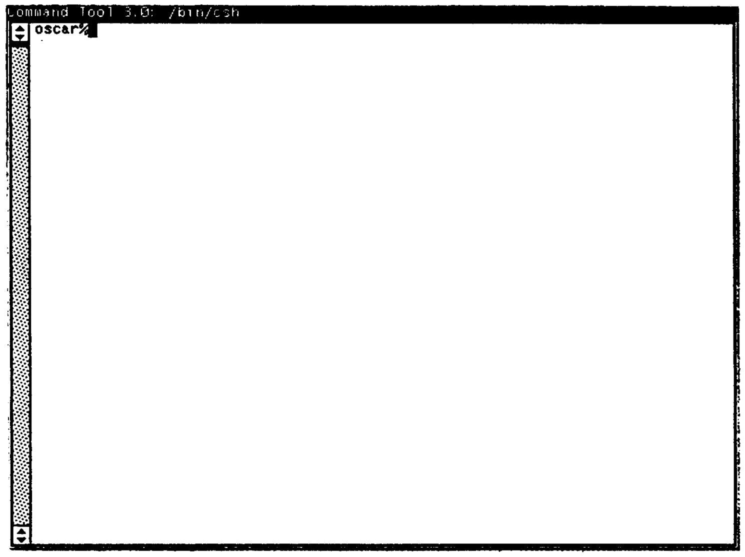
`cmdtool` is like `shelltool` with the added capability of the text facility. For example, you can type commands to the command prompt in `cmdtool`, just like with `shelltool`; also, you can scroll to previous screens of commands that you typed and search through the commands with [Find], just as in `textedit`.

To start `cmdtool`, pop up a root menu and choose the `CommandTool` item, or type `cmdtool &` to a shell command prompt.



`cmdtool` icon

Figure 10-1 `cmdtool`



```
oscar%
```

You can type commands to the command prompt, or you can move around, select text, search for a selection, and stuff from a file, as with `textedit`.

If you type:

```
ls /usr/medici/text
```

and then you notice that you forgot to include `/doc` before `/text`, you can position the caret in front of `/text` by pointing with the mouse and clicking, type `/doc`, then reposition the caret at the end of the line and type [RETURN].

In other words, the order in which you type the characters when composing a command is not what determines the characters the application receives. You can insert characters on and delete characters from the command line, using the text facility operations when you wish.

Then, when you type [RETURN], and the caret is at the end of the document, you send all of the characters between the start of the last command and the [RETURN] to the application, in this case the command interpreter.

The application does not see any of the characters that you type until you type [RETURN], at which point it receives all of the characters that comprise the command line.

10.2. Command Facility Characteristics

This section describes characteristics of the command facility.

Read-Only Boundary

By default, the command facility doesn't permit alteration of history — there is a read-only boundary prior to the current command line.

The `Edit On` text menu item in command subwindows permits you to remove the read-only boundary after starting a command subwindow. You can replace the boundary with `Edit Off` — the text menu alters itself according to whether or not you have a boundary.[27]

You can't insert into, or delete from command lines that the application has already executed. The Delete and Put function keys work only after the command prompt on the current command line. You can't move the caret within, or prior to the current command prompt.

The command facility notifies you that text is read-only when you attempt to delete it.

Interpretation of Certain Control Characters

When the caret is at the end of the document, a command subwindow interprets CTRL-C, CTRL-D, and CTRL-Z the same way as when you type them to a shell or `shelltool`.

When the caret is *not* at the end of the document, the command facility interprets two of these characters as a text subwindow would — it simply inserts any CTRL-C or CTRL-Z into the document.

When output appears in the command subwindow while you're typing, perhaps because you've typed ahead or you're running a process in the background, the command facility inserts the output in front of what you've typed to prevent interference with the current command line.

Moving to the Current Command Prompt

When you're scrolling through the text that includes older commands, you can move to the current command prompt at any time by typing CTRL-RETURN.

CTRL-RETURN moves you to the end of the text in `textedit` as well.

`vi`, `more`, and `su`: Cbreak or Raw Mode

For now, the command facility doesn't support the `vi` or `more` programs, or `man` because it uses `more`. You can use `cat` instead of `more` because you have scrolling capability to look back on previous portions of the output.

Also, a command subwindow echoes the passwords that you type in when using `su` or `rlogin` — so be careful that no one is looking when you type your password to `su` or `rlogin` in a command subwindow.[28]

[27] If you decide you don't like the read-only boundary default, you can remove it by setting the `Append_only_log` option to `False` in the `Text` category of `defaultsedit`.

[28] For information on the `rlogin` command, see *Using the Network: Beginner's Guide*. For information on the `su` command, see *Doing More With UNIX: Beginner's Guide*.

The reason for these problems is that the command facility cannot yet handle programs with cbreak or ''raw'' mode.

10.3. Summary

The command facility extends the concept of a command interpreter and a text editor to evolve a powerful tool for command execution and editing.

The window system also supports other editing tools, described in the next chapter.

11

Other Editing Tools

11

Other Editing Tools

Besides `textedit`, the Sun window system supports the editing tools `defaultsedit`, `iconedit`, and `fontedit`. You can set up your window system environment with `defaultsedit`, tailoring the window system to your personal needs. You can create ycur own icons for use within the window system using `iconedit`. Finally, you can design your own *fonts*, or typefaces, with `fontedit`.

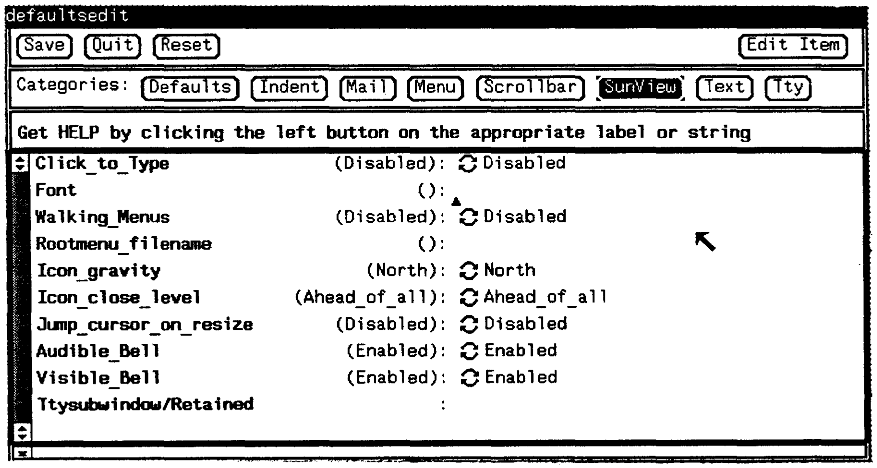
11.1. `defaultsedit`

`defaultsedit` permits you to tailor the window system and text facility to your preferences, for example, silencing the audible bell on your workstation if you desire.

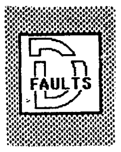
To start `defaultsedit`, pop up the root menu and choose the `DefaultsEditor` item, or type `defaultsedit &` to a command prompt.

Note: When you append the ampersand character (&) to a command line, you execute the command in the background, leaving the window free for typing other commands.

Figure 11-1 `defaultsedit`



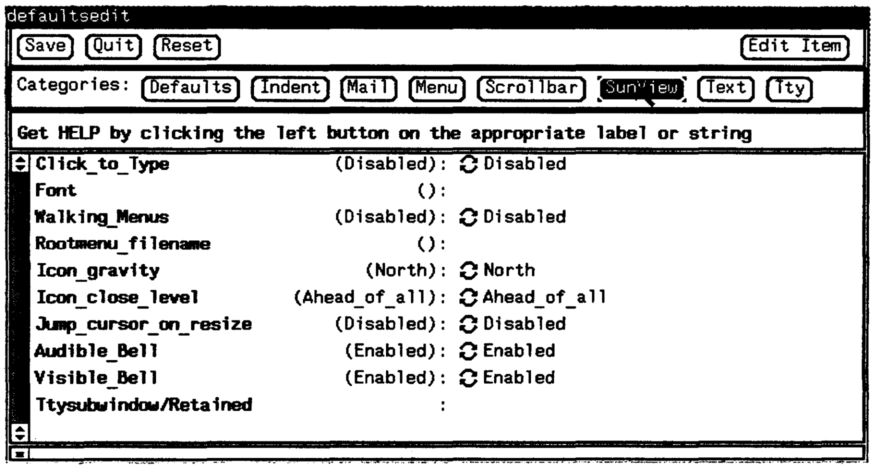
```
Get HELP by clicking the left button on the appropriate label or string
Click_to_Type                   (Disabled):  Disabled
Font                                    ():
Walking_Menus                   (Disabled):  Disabled
Rootmenu_filename                       ():
Icon_gravity                       (North):  North
Icon_close_level            (Ahead_of_all):  Ahead_of_all
Jump_cursor_on_resize           (Disabled):  Disabled
Audible_Bell                     (Enabled):  Enabled
Visible_Bell                     (Enabled):  Enabled
Ttysubwindow/Retained                     :
```



`defaultsedit` icon

To change a window system and text facility *option*, you must select the *category* of that option. For example, to change from old-style menus to new-style walking menus, select the `SunView` category.

Figure 11-2 `defaultsedit`*: Selecting a Category*

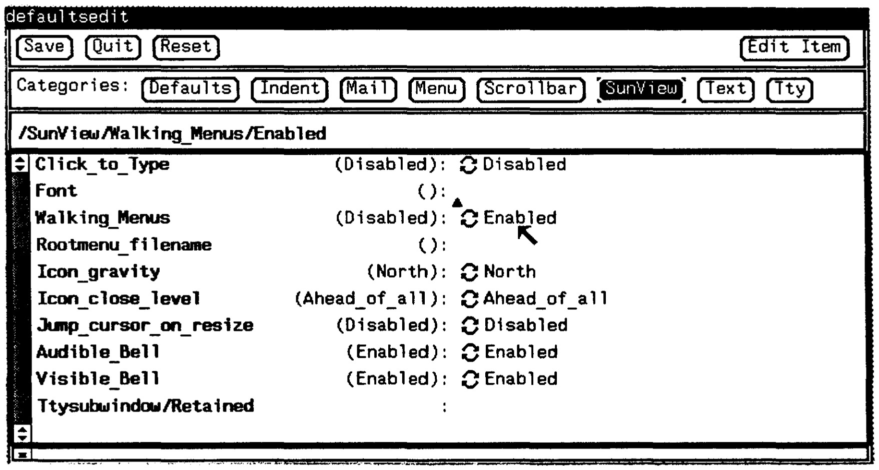


```
Get HELP by clicking the left button on the appropriate label or string
Click_to_Type                  (Disabled):  Disabled
Font                                   ():
Walking_Menus                  (Disabled):  Disabled
Rootmenu_filename                      ():
Icon_gravity                      (North):  North
Icon_close_level             (Ahead_of_all):  Ahead_of_all
Jump_cursor_on_resize          (Disabled):  Disabled
Audible_Bell                    (Enabled):  Enabled
Visible_Bell                    (Enabled):  Enabled
Ttysubwindow/Retained                    :
```

Now, you can explore each of the options and associated *settings* by clicking the left mouse button on the option name to get help information, or on one of the settings to pick that setting.

If you want to have walking menus, select the `Enabled` setting for the `Walking_Menus` option.

Figure 11-3 `defaultsedit`: *Selecting a Option Setting*



```
/SunView/Walking_Menus/Enabled
Click_to_Type                 (Disabled):  Disabled
Font                                 ():
Walking_Menus                 (Disabled):  Enabled
Rootmenu_filename                    ():
Icon_gravity                     (North):  North
Icon_close_level          (Ahead_of_all):  Ahead_of_all
Jump_cursor_on_resize         (Disabled):  Disabled
Audible_Bell                   (Enabled):  Enabled
Visible_Bell                   (Enabled):  Enabled
Ttysubwindow/Retained                   :
```

Now, your menus will appear in the new style.

`defaultsedit` supports several categories, including `SunView`, `Mail`, `Scrollbar`, `Text`, and `Menu`. The number of `defaultsedit` categories may change.

Explore each of the categories to see the variety of option settings available to you.

Some options require that you type in a setting, like `Font` which requires the filename where the font is stored. The filename for the font used to produce the screen illustrations in this manual is:

```
/usr/lib/fonts/fixedwidthfonts/screen.r.11
```

To save the options you have set, select the `Save` button. Whenever you set an option in the `SunView` category, you must exit and restart `suntools` for the new setting to take effect. Only the `SunView` category requires exiting and restarting to effect option setting changes.

To quit `defaultsedit`, select the `Quit` button.

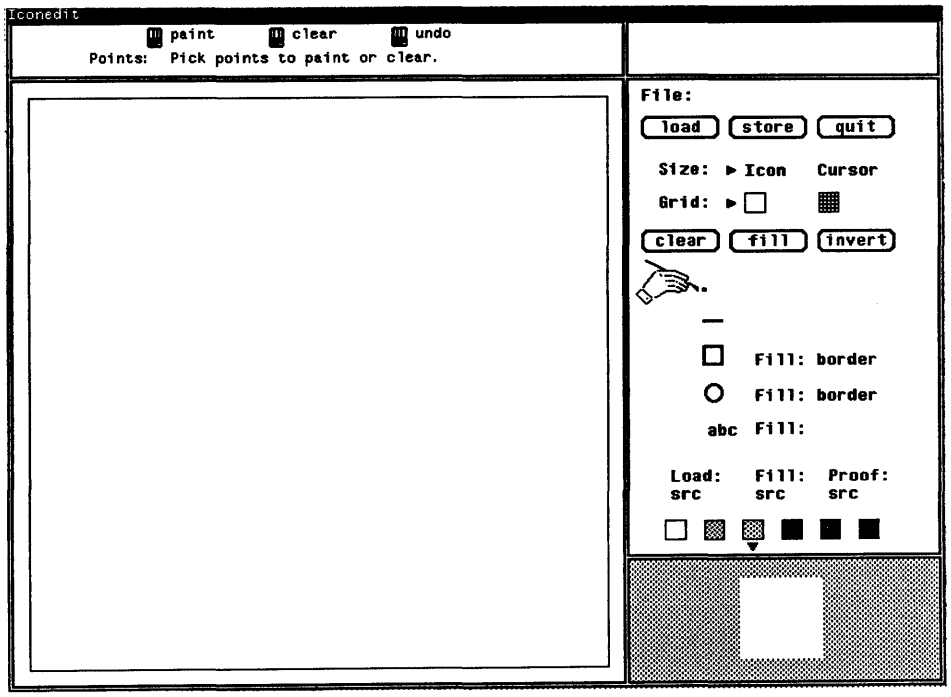
For more information on `defaultsedit`, see its Man Page, online or in the *Commands Reference Manual.*

11.2. iconedit

`iconedit` is an editing tool for icons and cursors. You can create new icons or cursors and store them in files, or edit old icons and cursors.

To start `iconedit`, choose the *IconEditor* item in the root menu, or type `iconedit &` to a command prompt.

Figure 11-4 `iconedit`



Points: Pick points to paint or clear.

Size: ▶ Icon Cursor

Grid: ▶ □ ▦

□ Fill: border

○ Fill: border

abc Fill:

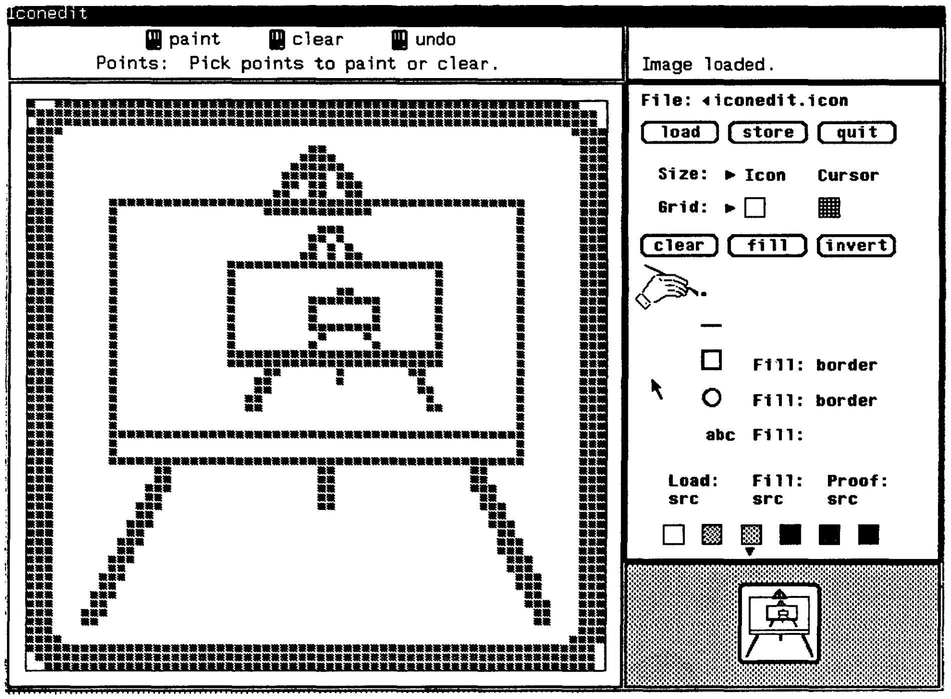
Load: src Fill: src Proof: src

Note: Creating and editing cursors is similar to creating and editing icons. Just remember to select `Cursor` as the setting for the `Size:` option, instead of the default `Icon` setting.

To create an icon from scratch, select points in the large subwindow. Notice how all the points appear in a miniature, ''actual size'' copy of the icon in the lower right corner.

Experiment by drawing points, lines, and filling rectangles or circles with various patterns. Invert the screen image by selecting the [invert] button, or clear the screen with [clear].

Figure 11-5 `iconedit` *Containing an Icon*



Points: Pick points to paint or clear.

Image loaded.

File: iconedit.icon

Size: ▶ Icon Cursor
Grid: ▶ □ ▦

Fill: border
Fill: border
abc Fill:

Load: src
Fill: src
Proof: src

Save your work by clicking near the `File:` marker, inserting the filename you wish to store, and selecting the [save] button.

To load a file, click near the `File:` marker, insert the filename you wish to load from, and select the [load] button.

To quit `iconedit`, select the [quit] button.

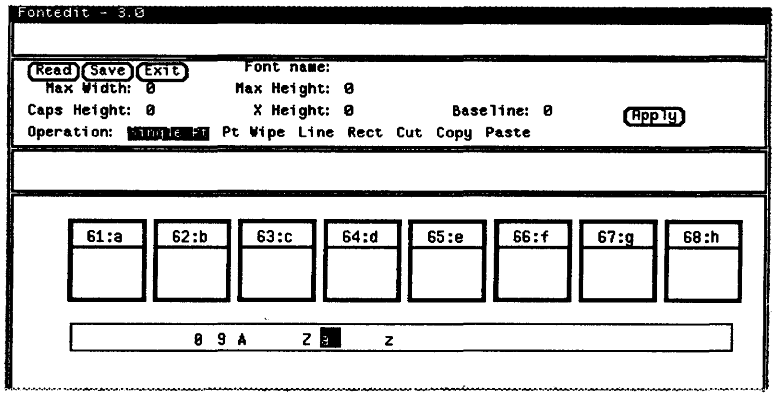
For more information on `iconedit`, see its Man Page, online or in the *Commands Reference Manual.*

11.3. `fontedit`

Use `fontedit` to design a font, a set of characters, or to alter an existing font.

To start `fontedit`, type `fontedit &` to a command prompt.

Figure 11-6 `fontedit`



```
Font name:
Max Width: 0          Max Height: 0
Caps Height: 0          X Height: 0          Baseline: 0
Operation: Single Pt  Pt Wipe Line Rect Cut Copy Paste
```

The top subwindow is the *message subwindow*, where `fontedit` displays relevant help and error messages.

Note: Experiment with `fontedit` on a copy of one of the standard font files, so that you don't lose the standard font by mistake.

Next, insert the filename of the font you want to edit after the `Font name:` marker. When you want to read in a font from that filename, select the [Read] button. You can experiment with the standard fonts for your machine, usually located in the `/usr/lib/fonts/fixedwidthfonts` directory.

When you want to save a font to that filename, select the [Save] button.

Below the message subwindow is the *control subwindow* for setting options.

Before you create a new font, you must set the following options:

- `Max Width:`
- `Max Height:`
- `Caps Height`
- `X Height` — height of the letter "x"
- `Baseline`

Select the [Apply] button to enter the options into `fontedit` before attempting to create any characters.

Next is the *proof subwindow* where you can type characters to see what they look like with the latest changes you have made.

The series of rectangles across the middle of `fontedit` allow you to select up to two characters for editing in the *editing area* below. Click on a character to make it appear below.

Click on the long rectangular scrollbar below the character selection area to change the series of selection rectangles to the appropriate portion of the alphanumeric font series. Also, you can type a character on the scrollbar, and it appears as the first character in the character selection area.

Note: You can select up to two characters to edit in the editing area at the same time.

Once you select a character, `fontedit` displays a magnified version and a proof version of the character in the editing area. Within the magnified version of the character in the editing area, you can edit the character by selecting any of a variety of operations listed after the `Operation:` marker in the control subwindow.

As mentioned above, you can type in the proof subwindow to check on your edits as you go along.

To save individual font characters, select the (Save) button next to the magnified version of the character in the editing area.

To save the entire font to the filename after the `Font name:` marker in the control subwindow, select the (Save) button in the control subwindow.

Exit by selecting the (Exit) button in the control subwindow.

For more information on `fontedit`, see its Man Page, online or in the *Commands Reference Manual.*

11.4. Summary

You've learned the basics of all of the editing tools, freeing you to explore the special-purpose displays in the next chapter.

12

Special-Purpose Displays

12

Special-Purpose Displays

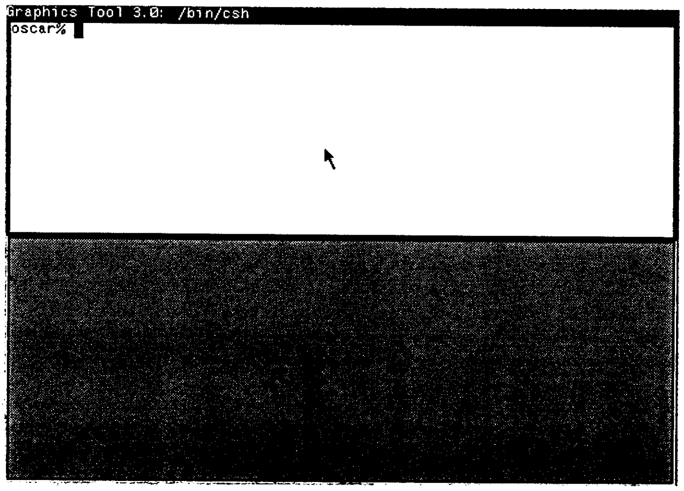
Some window system programs provide different types of displays. gfxtool divides a window in two parts; one for you to type commands, the other to view graphics programs. tektool is a Tektronix 4014 ™ *terminal emulator*, allowing you to run programs as if you are using a Tektronix 4014 terminal.

12.1. gfxtool

Note: You may want to start up gfxtool in the background of the parent window, so that you can continue to type commands there. Type the ampersand character (&) after gfxtool to run it in the background.

To start gfxtool, choose the GraphicsTool item on the root menu, or type gfxtool to the command prompt of a window. After a moment, a window with two parts, or *subwindows* appears. You can type commands to the upper subwindow, just like a shelltool. The lower subwindow, shaded dark gray, is a surface on which certain programs display graphics.

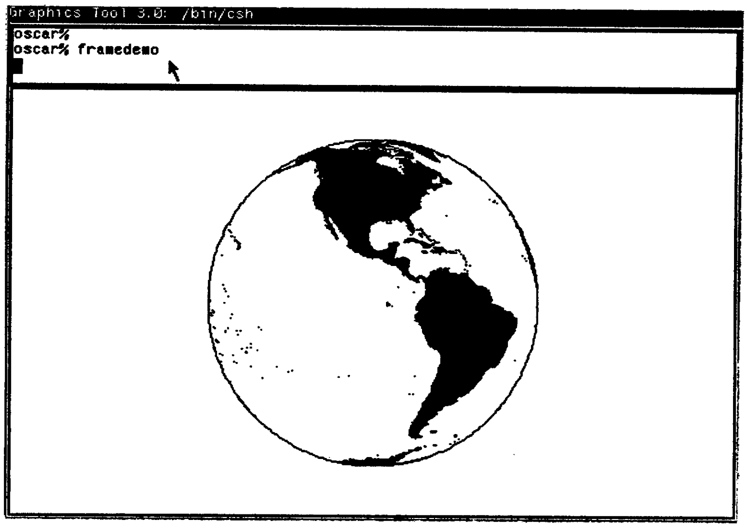
Figure 12-1 gfxtool



```
Graphics Tool 3.0: /bin/csh
oscar%
```

To run a graphics program with gfxtool, move the cursor into the upper subwindow and type the name of the program. This example shows the framedemo graphics demo.

Figure 12-2 `gfxtool` *Displaying the* `framedemo` *Graphics Program*



```
oscar%
oscar% framedemo
```

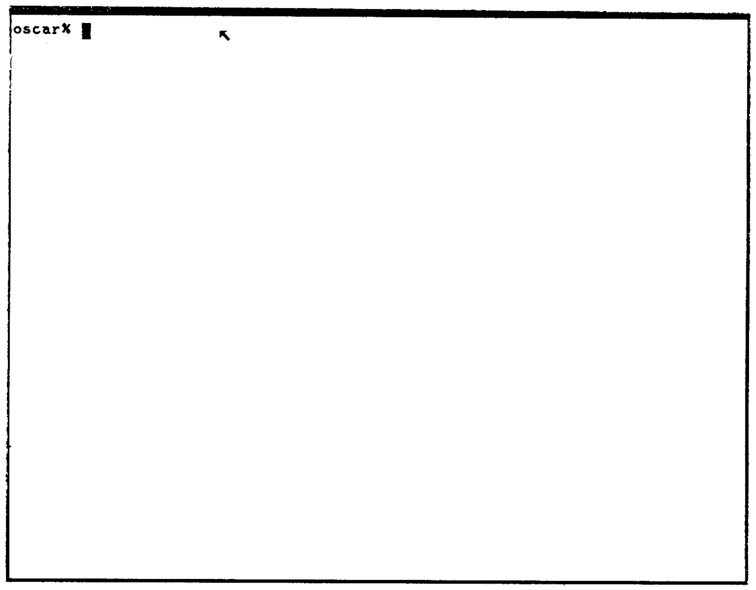
12.2. `tektool`

If you have used the Tektronix 4014 ™ terminal, you may want to take advantage of its features using your Sun Workstation.

You can start `tektool` just like `gfxtool`. Type `tektool` to a window command prompt. Use `tektool` just as you would use a Tektronix 4014 ™ terminal.

When a window appears, it will use the entire screen. The frame menu works as with any other tool.

Figure 12-3 `tektool`



```
oscar% 
```

For more information on `gfxtool` or `tektool`, see the Man Pages, online or in the *Commands Reference Manual.*

13

Menu Accelerators

13

Menu Accelerators

An *accelerator* is an abbreviated way to perform a text facility command or selected other SunView operations. One common accelerator is when you open an icon by clicking the left mouse button on it, instead of selecting the `Open` item on the application's frame menu.

This chapter describes:

- Accelerating to the end of a text subwindow
- Full-length zoom accelerators
- Mouse button accelerators

13.1. Accelerate to End of Text Subwindow

One important text facility accelerator is [CTRL-RETURN], which moves the caret to the end of the text in a text subwindow. This is especially useful when you want to get to the current command prompt in `cmdtool`.

13.2. Accelerated Full-Length Zoom

In addition to toggling full-length zoom of a window with the hidden `Zoom` item on a frame menu, you can hold down [CTRL] while typing the [Open] function key, in other words [CTRL-Open], as an alternative full-length zoom accelerator.

The next section describes another alternative — the mouse button accelerator [CTRL-LEFT] on the window frame.

13.3. Mouse Button Accelerators

Here is a table of mouse button accelerators, organized by function. This table and a table of accelerators organized by mouse button appear in the quick reference guide at the end of this manual.

Table 13-1 *Menu Accelerators Organized By Function*

Desired Function	***Mouse Button Procedure***
open	LEFT on icon
expose	LEFT on frame (border or namestripe)
hide	SHIFT-LEFT on frame (border or namestripe)
move	MIDDLE on corner = unconstrained, vertical and horizontal adjustment MIDDLE on middle third of edge = constrained, movement either horizontally or vertically
resize	CTRL-MIDDLE on corner = unconstrained, vertical and horizontal adjustment CTRL-MIDDLE on middle third of edge = constrained, resizing either horizontally or vertically
zoom full length	CTRL-LEFT on frame (border or namestripe) toggles (switches back and forth) this feature
menu access	RIGHT and hold down, on frame (border or namestripe)

14

Focusing Attention

14

Focusing Attention

Normally, when you use `suntools`, you choose the window you want to type in by moving the cursor inside the boundary box of that window. Then, whenever you type, the characters appear in that window, at least as long as you don't move the cursor. This is called the ''cursor-in-window'' or ''mouse-to-type'' model.

However, you can choose windows by an alternative method, called ''click-to-type.'' With ''click-to-type,'' you click the left or middle mouse button inside the window you want to type in, at a time when you aren't holding down any function keys. Then, no matter where you move the cursor, the characters you type will appear in the window that you chose.

14.1. Click-To-Type

To choose windows by ''click-to-type,'' rather than by moving the cursor inside a window boundary box, you must start `suntools` with the `-S` option, or specify ''click-to-type'' using `defaultsedit`.

Why Use ''Click-To-Type?''

This section describes two reasons why you might want to use ''click-to-type,'' rather than the standard method of choosing windows.

When you edit in a text subwindow of the text facility and you want to copy material from another text subwindow, you must be careful to press the [Get] key before exiting from the first subwindow, and remember to return to the cursor to the first window before you continue typing.

Similarly, while using `dbxtool`, you cannot move the cursor into the source window to select a location for a breakpoint, or move the cursor into the control panel to perform some operation, and then type a command without first having to return the cursor to the command subwindow. If you invoke an application that creates a window, from a `shelltool`, and then continue typing to your `shelltool` while waiting for the tool to finish initializing, the characters that you type to the shell may suddenly be directed to the tool whose window has just come up in an area that intersects the cursor.

How to Use "Click-To-Type"

An alternative to this user interface is the "click-to-type" model, in which the destination for mouse events and destination for keyboard events may be different. With this model, the destination for keyboard events is not changed until the user moves the mouse into another window and clicks the left (to choose the window and select a location) or middle (to choose a window while preserving the current location) mouse button. Thus, you can move the mouse outside of the window that you are editing, for example, to click buttons in a control panel, or to scroll or make a secondary selection in another window, and you don't have to return the mouse to the first window to type or edit there.

Specify your choice of the "cursor-in-window" or "click-to-type" model of user interface using `defaultsedit`. If you don't specify your choice, you will get "cursor-in-window."

Regardless of which user interface you employ, the visual feedback is the same: the border of the subwindow that has the keyboard focus darkens, and a blinking triangular or solid rectangular caret appears in the subwindow with the current focus of attention.

Carets in subwindows that don't have the current focus of attention are "shadow" carets, either gray diamonds or rectangular outlines.

15

Modifying Your Root Menu

15

Modifying Your Root Menu

When you want to construct a root menu other than the default root menu that `suntools` uses, you must:

- Construct a file with your root menu information
- Tell `defaultsedit` the name of the file you chose to store your root menu information
- Exit `suntools`
- Restart `suntools`

Later in the chapter, you can learn how to make a custom walking or stacking root menu, and in the last section, how to make a root menu with icons for the items you can pick.

15.1. Constructing a Root Menu File

The easiest way to construct a file containing your root menu information is to look at the file `suntools` uses to store its root menu information, and to glean what other information you need from your `.suntools` file or using the `toolplaces` command.

The Default Root Menu File

The file that `suntools` uses to store its root menu information is `/usr/lib/.rootmenu` which looks like this:

Figure 15-1 *Root Menu File for* suntools

```
venus% cat /usr/lib/.rootmenu
#
#   suntools root menu
#
"ShellTool"          shelltool
"CommandTool"        cmdtool
"MailTool"           mailtool
"TextEditor"         textedit
"DefaultsEditor"     defaultsedit
"IconEditor"         iconedit
"DbxTool"            dbxtool
"PerfMeter"          perfmeter
"GraphicsTool"       gfxtool
"Console"            shelltool -C
"Lock Screen"        lockscreen
"Redisplay All"      REFRESH
"Exit Suntools"      EXIT
venus%
```

One important point is that this file doesn't have any blank lines in it, especially not at the end of the file. `suntools` displays an error when it see a blank line in a root menu file.

However, you can separate the two columns with tab characters.

In the left column of the file, surrounded by quotation marks, are the labels for the items that appear on the root menu. In the right column are the programs that you run when you pick each menu item.

For example, the first label and program name are `"Shelltool"` and `shelltool`. The label, `"Shelltool"` appears without quotation marks on the actual root menu and, when you pick that item, you start a `shelltool`.

The program next to the `"Console"` label is `shelltool -C`, which runs a modified shelltool called a *console* window. You shouldn't run more than one console window at a time.

You can put options after the program name, and commands to run within the program, just as when you construct your own `.suntools` file.

Finally, `REFRESH` and `EXIT` are two special indicators to `suntools` that you want to refresh the entire screen, or to exit from `suntools`.

Make Your Own Root Menu File

By looking at your `.suntools` file, and running `toolplaces` when you like the way you've set up the windows on your screen, you can come up with a set of window attributes for each window you want to pick from your own root menu.

Note: Remember to make sure not to put any blank lines in the root menu file, especially not at the end of the file.

Put all of those attributes into a file in your home directory, designing the file after the example of the standard `/usr/lib/.rootmenu` file, and give the file the traditional `.rootmenu` filename, so you can remember where you've put your root menu information.

15.2. Inform `defaultsedit` of Your Filename

To inform SunView that you want to use your own root menu file:

- Start `defaultsedit`[29]
- Select the `SunView` category
- Select the `Rootmenu_filename` option

Note: Once you have set your root menu filename with `defaultsedit`, you won't have to exit and restart `suntools` to see the effects of changes you've made to your root menu file.

- Type the name of the filename you desire as the setting for the `Rootmenu_filename` option
- Select the (Save) panel button in `defaultsedit.`
- Quit `defaultsedit` by selecting the (Quit) panel button.

15.3. Exit and Restart `suntools`

After you exit and restart `suntools`, and you pop up the root menu, your new root menu should appear.

15.4. Walking or Stacking Root Menus

You can construct a walking or stacking root menu. Depending on how you have set the `Walking_Menus` option in the `SunView` category of `defaultsedit`, the menu will appear as a new-style walking menu or as an old-style stacking menu.

The walking menu configuration stores the hidden menu items in pull-right menus; whereas the stacking menus hides them in the lower overlapping menus.

All you have to do is this — construct a root menu file that refers to root menu extension files. Here is an example:

Figure 15-2 *A Root Menu File That Refers to Extension Files*

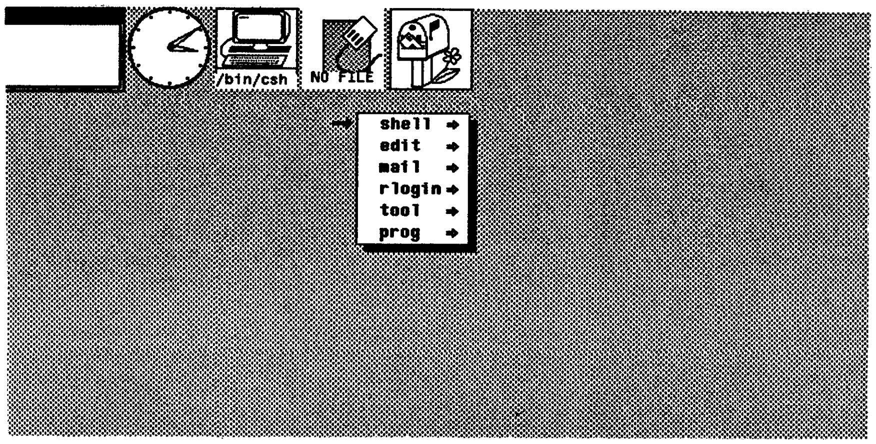
```
#
#       suntools root menu
#
shell       MENU     .rootmenu.shell
edit        MENU     .rootmenu.edit
mail        MENU     .rootmenu.mail
rlogin      MENU     .rootmenu.rlogin
tool        MENU     .rootmenu.tool
prog        MENU     .rootmenu.prog
```

Put the character string `MENU` in the program location of the right column, followed by a tab character and the filename of the root menu extension file.

[29] For more information on `defaultsedit`, see Section 11.1 .

On your screen, the extended root menu looks like this:

Figure 15-3 *The Extended Root Menu*



Here is an example of one of the root menu extension files:

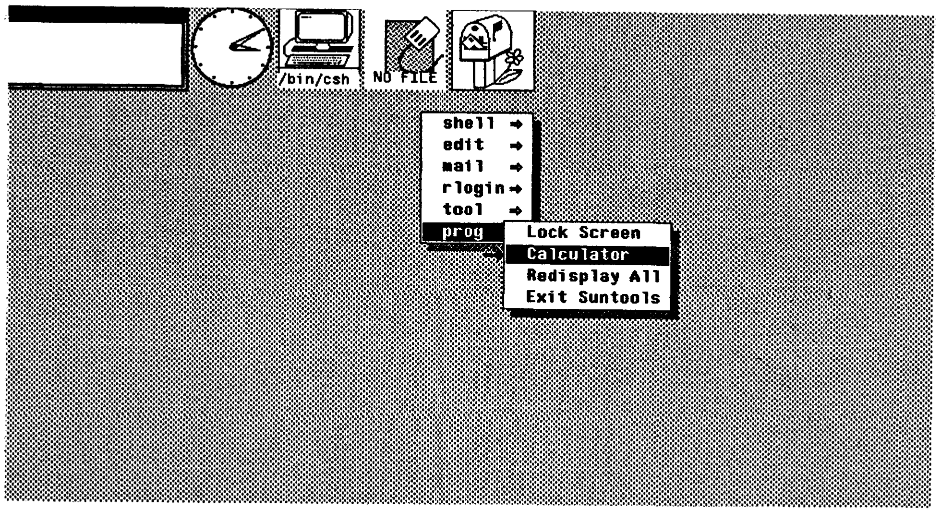
Figure 15-4 *A Root Menu Extension File*

Note: The lines beginning with the -WP option have wrapped around the end of the screen, and are actually part of the previous line.

```
venus% cat .rootmenu.shell
#
#       shell ".rootmenu" extension file
#
"ShellTool"             shelltool   -Wp    0   66 -Ws 570 715
 -WP  640   0
"CommandTool"           cmdtool
"Console"               cmdtool     -Wp    0    0 -Ws 570  67
 -WP    0    0 -Wl "<< CONSOLE >>" -WL console -C
venus%
```

Here is what the root menu extension file produces when you've set SunView to the new-style walking menu:

Figure 15-5 *Pull-Right Menu from Custom Root Menu*



15.5. Using Icons to Illustrate Your Root Menu

Now that you've extended your root menu by replacing the right column of programs with additional menus, you can illustrate your root menu by replacing the left column of labels with icons.

Shopping for Icons

You can pick from the icons stored in the `/usr/include/images` directory. All of the files with filenames ending in `.icon`, such as `lockscreen.icon`, `nautilus.icon`, and `terminal.icon`, are icons.

Inserting the Icons into the Root Menu

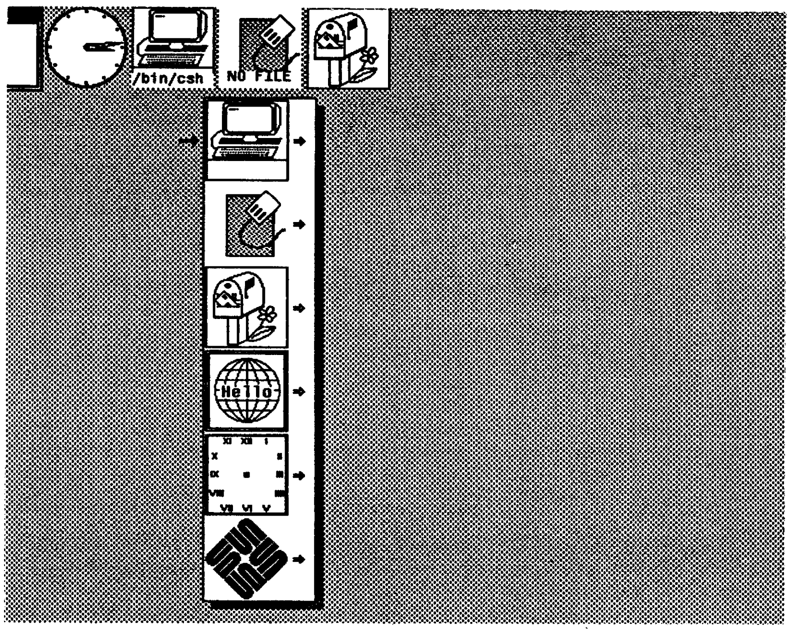
Replace the left column of labels, surrounded by quotation marks, with a new column of icon filenames surrounded by a less than character (<) on the left and a greater than character (>) on the right. Here is the extended root menu file with icons, rather than labels:

Figure 15-6 *Root Menu File With Icons*

```
venus% cd
venus% cat .rootmenu
#
#       suntools root menu
#
</usr/include/images/terminal.icon>      MENU   .rootmenu.shell
</usr/include/images/textedit.icon>      MENU   .rootmenu.edit
</usr/include/images/mail.icon>          MENU   .rootmenu.mail
</usr/include/images/hello_world.icon>   MENU   .rootmenu.rlogin
</usr/include/images/clock.rom.icon>     MENU   .rootmenu.tool
</usr/include/images/lockscreen.icon>    MENU   .rootmenu.prog
venus%
```

This is what the illustrated and extended root menu looks like:

Figure 15-7 *Illustrated Root Menu*



Now, you know how to modify your root menu.

16

Modifying Subwindow Behavior

16

Modifying Subwindow Behavior

You may want to modify the behavior of your subwindows, for example to assign new functions or filters to keys. To modify a subwindow in this way, you put a particular file in your home directory. For tty subwindows like `shelltool`, use the `.ttyswrc` file; for text subwindows like `textedit`, use the `.textswrc` file.

Be careful about trying to assign new functions to the function keys used by the text facility or to the arrow keys. When you start up a subwindow within `suntools`, the text facility assumes control of your text facility function keys, disallowing any other functions you may have assigned to those keys.[30]

When your subwindow displays:

`.ttyswrc error:` *keyname* `cannot be mapped.`

it means that your subwindow read a `.ttyswrc` file that specified key functions that conflict with the text facility function key assignments.

The following sections describe modifying tty and text subwindow behavior, including an explanation of how to map functions to the arrow keys.

16.1. Modifying Tty Subwindow Behavior

When you put a `.ttyswrc` file in your home directory, you can *map*, or assign, alternate functions to keys and set on or off the *page mode*, or scrolling behavior, of your tty subwindows.

Page Mode On/Off

Page mode, with settings on or off, describes the scrolling behavior of the tty subwindow.

When you type `cat` *filename* to look at a long file, the text scrolls by on a tty subwindow, so you probably miss what you were looking for in the file. When the file scrolls by like this, page mode is **off**.

However, you can turn page mode on by picking the `Page Mode On` item on the *tty menu*, accessible by pressing the right mouse button while the cursor is inside the tty subwindow.

[30] For information on setting up your function keys, see Section 8.1 .

When page mode is on, text scrolls by one page at a time with a tiny stop sign appearing on the screen at the end of each page. When you want to see the next page, type any character, or select the `Continue` item in the tty menu.

By default, page mode is off in tty subwindows. However, you can set the default to page mode on with a specification in your `.ttyswrc` file described below.

Mapping Functions to Keys

Several window applications provide a *terminal emulator*, or a program that runs a command interpreter subwindow, or shell subwindow, in some part of the application. When you type to this subwindow, the application passes the keystrokes to the program running in it, rather handling them itself. Output from programs run in this way displays in the subwindow.

Whenever you start up a tty subwindow, it looks for a `.ttyswrc` file in your home directory, and reads it if the file is there. In that file, you can map new functions to keys, in effect creating your own terminal emulator.[31]

Format of the `.ttyswrc` File

The format of the `.ttyswrc` file is:

#	Comments.
`set` *variable*	Turn on the specified variable.
`mapi` *key text*	When *key* is typed pretend *text* was typed.
`mapo` *key text*	When *key* is typed pretend *text* was output.

The only currently defined *variable* is `pagemode`. A *key* must be one of: L1 through L15, F1 through F15, T1 through T15, R1 through R15, and LEFT or RIGHT. *Text* may contain *escapes*, such as `\E` for escape, `\n` for newline, and `^X` for CTRL-X.[32]

It is possible for a terminal-based program to send special escape sequences to an application to perform certain operations. These escape sequences may also be sent using the `mapo` function described above. You can send the following functions to the window application in which the tty subwindow resides, not to the tty subwindow itself.

`\E[1t`	– open
`\E[2t`	– close (become iconic)
`\E[3t`	– move, with interactive feedback
`\E[3;TOP;LEFTt`	– move, to `TOP LEFT` (pixel coordinates)
`\E[4t`	– stretch, with interactive feedback
`\E[4;WIDTH;HTt`	– stretch, to `WIDTH HT` size (in pixels)
`\E[5t`	– top (expose)
`\E[6t`	– bottom (hide)

[31] When using the default kernel keyboard tables, you can't map the keys L1, LEFT, RIGHT, BREAK, R8, R10, R12, and R14 in this way because they send special values to the terminal emulator subwindow. See the `kbd` Man Page, Section 5 of the Man Pages, for more information.

[32] See the `termcap` Man Page, Section 5 of Man Pages, online for the format of the string escapes that the tty subwindow recognizes.

`\E[7t`	– refresh
`\E[8;ROWS;COLSt`	– stretch, to `ROWS COLS` size (in characters)
`\E[11t`	– report if open or iconic by sending `\E[1t` or `\E[2t`
`\E[13t`	– report position by sending `\E[3;TOP;LEFTt`
`\E[14t`	– report size in pixels by sending `\E[4;WIDTH;HTt`
`\E[18t`	– report size in characters by sending `\E[8;ROWS;COLSt`
`\E[20t`	– report icon label by sending `\E]Llabel\E\`
`\E[21t`	– report tool header by sending `\E]llabel\E\`
`\E]l`*<text>*`\E\`	– set tool header to *<text>*
`\E]I`*<file>*`\E\`	– set icon to the icon contained in *<file>*; *<file>* must be in `iconedit` output format
`\E]L`*<label>*`\E\`	– set icon label to *<label>*
`\E[>OPT;...h`	– turn `OPT` on (`OPT` = 1 => pagemode), e.g., `\E[>1;3;4h`
`\E[>OPT;...k`	– report `OPT`; sends `\E[>OPTl` or `\E[>OPTh` for each `OPT`
`\E[>OPT;...l`	– turn `OPT` off (`OPT` = 1 => pagemode), e.g., `\E[>1;3;4l`

As an example of the use of this facility, you can put these aliases into your `~/.cshrc` file:

```
# dynamically set the name stripe of the tool:
alias header 'echo -n "^[]l\!*^[\"'
# dynamically set the label on the icon:
alias iheader 'echo -n "^[]L\!*^[\"'
# dynamically set the image on the icon:
alias icon 'echo -n "^[]I\!*^[\"'
```

The next section demonstrates an example of a `.ttyswrc` file into which you can insert these function mappings.

Example .ttyswrc File

Here is an example .ttyswrc file that you can copy into your home directory:

Figure 16-1 *Example* .ttyswrc *File*

```
venus% cat .ttyswrc
#
#     ttysubwindow startup file
#
set     pagemode
# Top:
mapo     T1     E[5t
# Close:
mapo     T2     E[2t
# Move:
mapo     T3     E[3t
# Stretch:
mapo     T4     E[4t
# Bottom:
mapo     T5     E[6t
# Refresh:
mapo     T6     E[7t
# Move (non-interactive) to top left:
mapo     T7     E[3;1;1t
# Stretch in chars (non-interactive) to half high:
mapo     T8     E[8;10;80t
# Stretch in chars (non-interactive) to normal size:
mapo     T9     E[8;34;80t
# Commands (very left keys, not setup):
mapi     L3     jobs
mapi     L5     mail
mapi     L7     ls -F
mapi     L9     more errs
# Editing:
# Bracket word in italic escapes while in vi:
mapi     R4     iEeaE
# Bracket word in bold escapes while in vi:
mapi     R5     iEeaE
# Mail:
mapi     R1     dt
mapi     R2     s +planning\ndt\n
mapi     R3     s +bugs\ndt\n.sp
venus%
```

Of course, you can modify this file to create a .ttyswrc file of your own.

16.2. Modifying Text Subwindow Behavior

Each time you create a text subwindow, it looks for a `.textswrc` file in your home directory and reads it if the file is there.

Mapping Filters or Functions to Keys

The text facility provides the ability to pipe the primary selection to an arbitrary program, or filter (as `stdin`), and insert the resulting output (`stdout`) at the caret. If the primary selection is in pending-delete, the output from the filter will replace it.

For example, piping a selection through `fmt` will cause all of the long lines to be truncated to 72 characters. Piping a selection through `indent` will format it according to the defaults specified in `~/.indent.pro`[33]

You can bind filter to unused function keys, in other words, to [R1] through [R14], [F1] through [F9], or when you have mapped the standard editing operations to the function keys on the right side of the keyboard, to [L1] through [L10].

Format of the `.textswrc` File

Here is the basic format of each entry:

key-name *FILTER*
command-line

For example,

```
KEY_RIGHT(14)    FILTER
fmt
```

causes [R14] to pipe the current selection through `fmt`, and

```
KEY_RIGHT(11)    FILTER
indent -st
```

causes [R11] to format the current selection.[34]

You can find the following useful filters in `/usr/bin`:

`insert_brackets` *arg1 arg2*
: Copies arg1 to *stdout*, then copies *stdin* to *stdout*, then copies arg2 to *stdout*, in general, `insert_brackets ( )` can be used as a filter for parenthesizing the selected text. If the selected text already has the indicated brackets, then insert_brackets removes them.

[33] See the `indent` Man Page, online or in the *Commands Reference Manual*.

[34] `s` and `t` are options to `indent` that tell it to read from *stdin* and output to *stdout*.
The program you specify must appear in a directory accessible by your search path. If the program is not found, or there is some other failure in the invocation of the filter, no change will occur in the document, even if the primary selection is pending-delete.

`shift_lines` *n*	Moves the selected text right *n* spaces (left when *n* is negative).
`capitalize`	Capitalize/uncapitalize the selected material, as follows: If everything is in caps, convert everything to lower case, for example, `EDIT TOOL` becomes `edit tool`. If the selected material consists of several words separated by white space, then for each word, capitalize it if it isn't, lower case if it is, for example: `now is the time` becomes `Now Is The Time`, and vice versa. Finally, for material consisting of a sequence of one or more letters without any white space, if there are any lower case letters, convert everything to uppercase, for example: `adjust_pending_delete` becomes `ADJUST_PENDING_DELETE`.

The file `/usr/lib/.textswrc` contains the following bindings for the right hand function keys:

[R1] Italics

[R2] Bold

[R3] Listing font

[R4] Capitalize

[R5] Parenthesize

[R6] Insert/remove

[R7] 2 points smaller

[R10] Shiftlines left 1 tab

[R11] Indent

[R12] Shiftlines right 1 tab

[R14] `fmt`

[R15] Insert current time and date

Example .textswrc File

Here is an example .textswrc file located in /usr/lib/.textswrc:

Figure 16-2 *Example* .textswrc *File*

```
venus% cat /usr/lib/.textswrc
/*   textswrc file        */

KEY_RIGHT(1) FILTER
insert_brackets
/*
 * insert_brackets inserts the indicated characters around the selected text.
 * If the text is already bracketed with these characters, it removes them.
 * At some point in the future, the remove option will be invoked by holding
 * down SHIFT key while hitting function key.
 */

KEY_RIGHT(2) FILTER
insert_brackets

KEY_RIGHT(3) FILTER
insert_brackets

KEY_RIGHT(4) FILTER
capitalize
/*
 * capitalize/uncapitalize selected material as follows: If there are no
 * lower case characters, convert everything to lower case.
 * Example: EDIT TOOL => edit tool.
 * If there are any lower case letters, convert everything to uppercase.
 * Example: adjust_pending_delete => ADJUST_PENDING_DELETE
 *
 * If characters consist of several words separated by white space, then for each
 * word, capitalize the first letter it if it isn't, lower case if it is.
 * Example: now is the time => Now Is The Time, and vice versa.
 */

KEY_RIGHT(5) FILTER
insert_brackets ( )

KEY_RIGHT(6) FILTER
insert_brackets " "

KEY_RIGHT(7) FILTER
insert_brackets
```

Continued on next page

Continued from previous page

```
KEY_RIGHT(10) FILTER
shift_lines -t -1
/*
 * shifts selected lines left the indicated number of tab stops tab stops
 * computed the following way: if any spaces are seen on the first line tab
 * stops = 4. Otherwise, look in .indent.pro for corresponding value (you can
 * use the indent_tool to set up your .indent.pro). If no .indent.pro, assume
 * 8. Note this means that if the user has that user can specify 8 in his
 * .indent.pro, but for those files that he has explitily formatted using 4
 * space indentation this will do the right thing provided that the first
 * line given to shift_lines contains some spaces.
 *
 */

KEY_RIGHT(12) FILTER
shift_lines -t 1
/* shifts selected lines right the indicated number of spaces */

KEY_RIGHT(11) FILTER
indent -st
/* invoke indent on the selected material, using the options specified in
 * .indent.pro
 */


KEY_RIGHT(13) FILTER
whenis -nw now
/* inserts current time */

KEY_RIGHT(14) FILTER
fmt
/* pass the selected material through fmt */


venus%
```

As with the `.ttyswrc` file, you can modify this example file to create a `.textswrc` file of your own. **Caution** — read the next section before copying this example file into your home directory.

Mapping Functions to the Arrow Keys

If you absolutely have to map functions to your arrow keys, as in the example `.textswrc` above, you must add:

```
setkeys noarrows
```

on a line by itself in your `.login` file, so SunView won't reassign the arrow keys.

17

Other Features

17

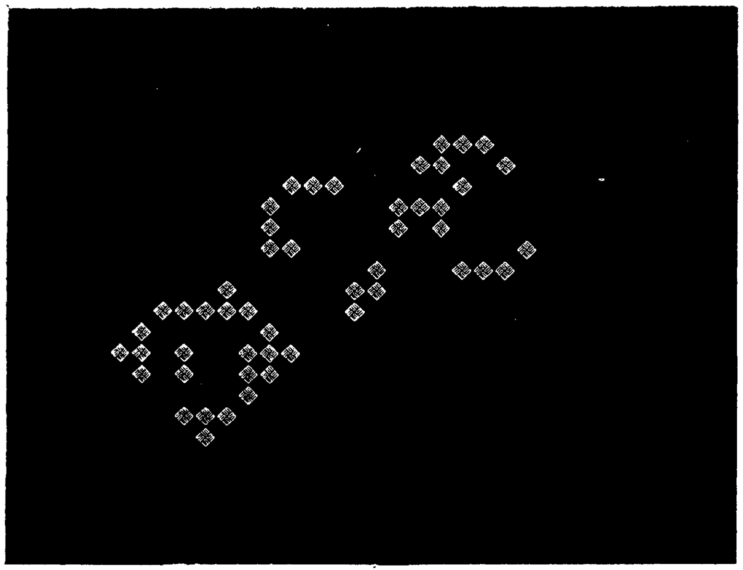
Other Features

Two other useful features of the window system are `lockscreen` and `screendump`. `lockscreen` locks your screen and displays the `life` graphics demo when you want to stop work for a while. `screendump` takes a picture of everything on the screen, so that you can save the image, or print it out. `screendump` was used to produce most of the pictures in this document.

17.1. `lockscreen`

To activate `lockscreen`, choose the `Lock Screen` menu item on the root menu. `lockscreen` clears the screen, then displays the `life` graphics demo, restricting access to your machine.

Figure 17-1 `lockscreen` *Displays the* `life` *Graphics Demo*



Note: When the system crashes for some reason during a time when `lockscreen` is running, then the system will probably reboot itself, leaving the system login prompt, without any graphics demo, on the screen.

To start work again, click the left mouse button to clear the `life` program display from your screen. Then, `lockscreen` prompts you for a password. If you type your password correctly, `lockscreen` displays your windows on the screen, identical to when you locked the screen.

To learn more about `lockscreen`, such as how to specify other graphics demo displays, see the `lockscreen` Man Page, online or in the *Commands Reference Manual.*

17.2. `screendump`

`screendump` allows you to take a snapshot of the screen at a given time and store it in a file or print it out on certain printers. Simply type `screendump >` *filename* to save the screen image in a file.

You may want to `rlogin` to your workstation from another machine, so that when you type `screendump` to your workstation, your command line doesn't appear in the screen image. See *Using the Network: Beginner's Guide* for more information about `rlogin`.

For more information about `screendump`, see its Man Page, online or in the *Commands Reference Manual.*

A

Further Reading

A

Further Reading

When you want to read more, start with these manuals:

Mail and Messages: Beginner's Guide
Games, Demos, and Other Pursuits: Beginner's Guide
Doing More With UNIX: Beginner's Guide
Using the Network: Beginner's Guide

SunView Programmer's Guide
SunView System Programmer's Guide
The Pixrect Reference Manual
Commands Reference Manual
System Administration for the Sun Workstation

B

Glossary

B

Glossary

This glossary lists SunView terms in common use, especially in this manual. The quick reference, located at the end of this manual provides a quick reference for the mouse buttons and the function keys.

adjusting a selection
Moving the cursor and clicking to make a selection encompass more or fewer characters.

caret
A blinking triangle, gray diamond, static solid or ''shadow'' rectangle shape that indicates the insertion point in a text subwindow.

click
Press and release a mouse button.

click-to-type
One method for choosing the window to which what you type on the keyboard will go — click the left or middle mouse button while the cursor is within a window border to focus attention on that window. Until you click again while the cursor is within another window border, everything you type on the keyboard will be sent to the window that has the current focus of attention.

command subwindow
A text subwindow that includes a command interpreter, like `cmdtool`.

cursor
The distinctive shape, usually an arrow pointing ''north-west,'' that follows and indicates the position of the mouse.

cursor-in-window
One method for choosing the window to which what you type on the keyboard will go — move the cursor within a window border to focus attention on that window.

current selection
The same as **primary selection**.

focus of attention
The window to which characters typed on the keyboard will be sent, picked by a cursor with either the ''cursor-in-window'' or ''click-to-type'' model.

frame
The border and namestripe surrounding a subwindow.

frame menu
The menu you obtain by holding down RIGHT over the border or namestripe of any window. Contains the operations `Close, Move, Resize, Expose, Hide, Redisplay,` and `Quit.`

insertion point
The place where the next character you insert will go, usually at the caret.

inverse video
Reversed light and dark portions of the screen.

LEFT Left button on mouse.

menu
A list of operations that you can choose by clicking RIGHT and moving the cursor. Menus appear when, and for as long as, you hold down RIGHT. The operation occurs when you release RIGHT with the cursor over a menu item.

menu item
The area of a menu that you choose to invoke a particular operation.

mouse-to-type
Same as **cursor-in-window.**

new-style menu
The same as **walking menu.**

panel
A subwindow with buttons that acts like a control panel for an application, including certain text facility editing operations.

pending delete selection
A selection, either primary or secondary, made while the CTRL key is down. Pending delete selections are automatically deleted by a subsequent insertion. Pending delete selections are indicated by a gray overlay.

MIDDLE Middle button on mouse.

point at
Position the cursor over.

primary selection
The span of text highlighted by inverse video, and marking the area for current editing operations. Also called the current selection.

pull-right menu
A menu accessible by moving the mouse to the arrow (=>) at the right of an item on a new-style walking menu.

RIGHT Right button on mouse.

root menu
The menu you obtain by holding down RIGHT with the cursor over the

gray background area of the screen.

scroll bar
A gray column at the left of the text subwindow used for scrolling and indicating position within the entire text.

scroll button
The box at the top or bottom of the scroll bar used for scrolling.

scrolling
Moving the text up or down in a subwindow; can also think of it as repositioning the subwindow with respect to the data it displays.

secondary selection
A selection made while a function key is held down. The selection only lasts as long as the function key is held down, and is indicated by underlining the selected characters. These characters will be an operand to the corresponding function.

select
Indicate a span, a contiguous sequence of characters, by pointing at and clicking.

selection
A span of characters, highlighted by inverse video, underlining, or gray shading.

shadow caret
A gray diamond or rectangular outline shape that indicates the insertion point within a window that isn't the current focus of attention.

shelf
A place where the text facility stores the text you [Put] or [Delete], using function keys.

span
A contiguous sequence of characters.

stacking menu
A menu with overlapping parts accessible by clicking the left mouse button while continuing to hold the right mouse button that popped up the menu.

text menu
The menu you obtain by holding down [RIGHT] anywhere inside a text subwindow. Contains operations such as `Reset`, `Save`, `Set Directory`.

text subwindow
A subwindow that provides display and editing capability for text within the text facility.

tiled windows
Windows that do not overlap, forming a pattern like pieces of a mosaic.

tty subwindow
A subwindow that includes a command interpreter, but doesn't support all of

the text facility operations, for example, `shelltool`.

walking menu
A new style of menu that contains pull-right menus accessible by moving the mouse to the arrow (=>) at the right of an item on the main menu.

Index

Revision History

Version	Date	Comments
A	17 February 1986	3.0 Release. Rework of **Using the Shell** section of 2.0 Release of the *Beginner's Guide to the Sun Workstation.*

Windows and Window-Based Tools: Quick Reference

This quick reference describes the text facility operations of the mouse and the function keys.

The Mouse

Each of the mouse buttons has a general purpose:

LEFT	*selects*
MIDDLE	*adjusts* (or m*odifies*) a selection
RIGHT	pops up a menu

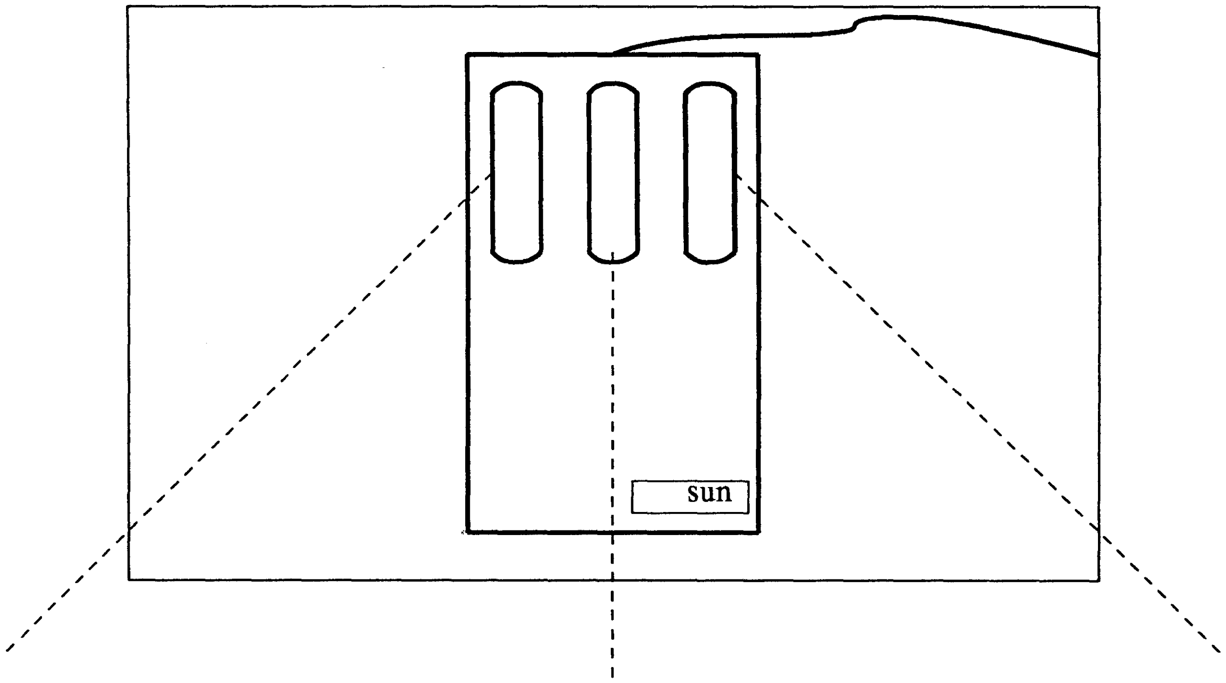
The SHIFT key toggles the ''direction'' of the action.

The CTRL key modifies the meaning of a choice or accelerator.

Mouse By Function

Desired Function	*Mouse Button Procedure*
open	LEFT on icon
expose	LEFT on frame (border or namestripe)
hide	SHIFT-LEFT on frame (border or namestripe)
move	MIDDLE on corner = unconstrained, vertical **and** horizontal adjustment MIDDLE on middle third of edge = constrained, movement either horizontally **or** vertically
resize	CTRL-MIDDLE on corner = unconstrained, vertical **and** horizontal adjustment CTRL-MIDDLE on middle third of edge = constrained, resizing either horizontally **or** vertically
zoom full length	CTRL-LEFT on frame (border or namestripe) toggles (switches back and forth) this feature
menu access	RIGHT and hold down, on frame (border or namestripe)

Mouse By Button



MIDDLE	On corner = unconstrained move, vertical **and** horizontal adjustment On middle third of edge = constrained move, either horizontally **or** vertically
SHIFT-MIDDLE	n.a.
CTRL-MIDDLE	On corner = unconstrained resize, vertical **and** horizontal adjustment On middle third of edge = constrained resize, either horizontally **or** vertically

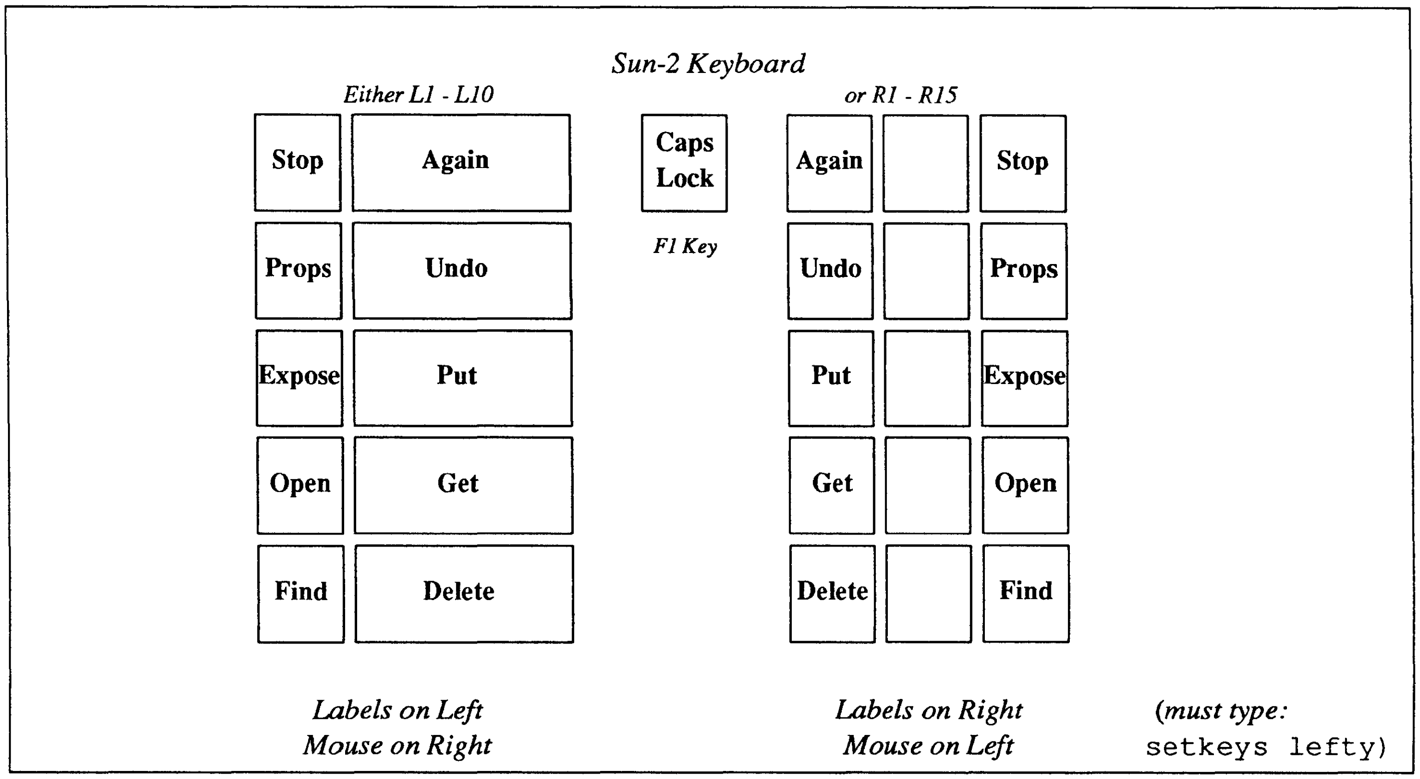
LEFT	On icon: open On frame (border/namestripe): expose
SHIFT-LEFT	On frame (border/namestripe): hide
CTRL-LEFT	On frame (border/namestripe): toggles zoom (full length)

RIGHT or SHIFT-RIGHT or CTRL-RIGHT	On icon or frame (border/namestripe): pop up a menu

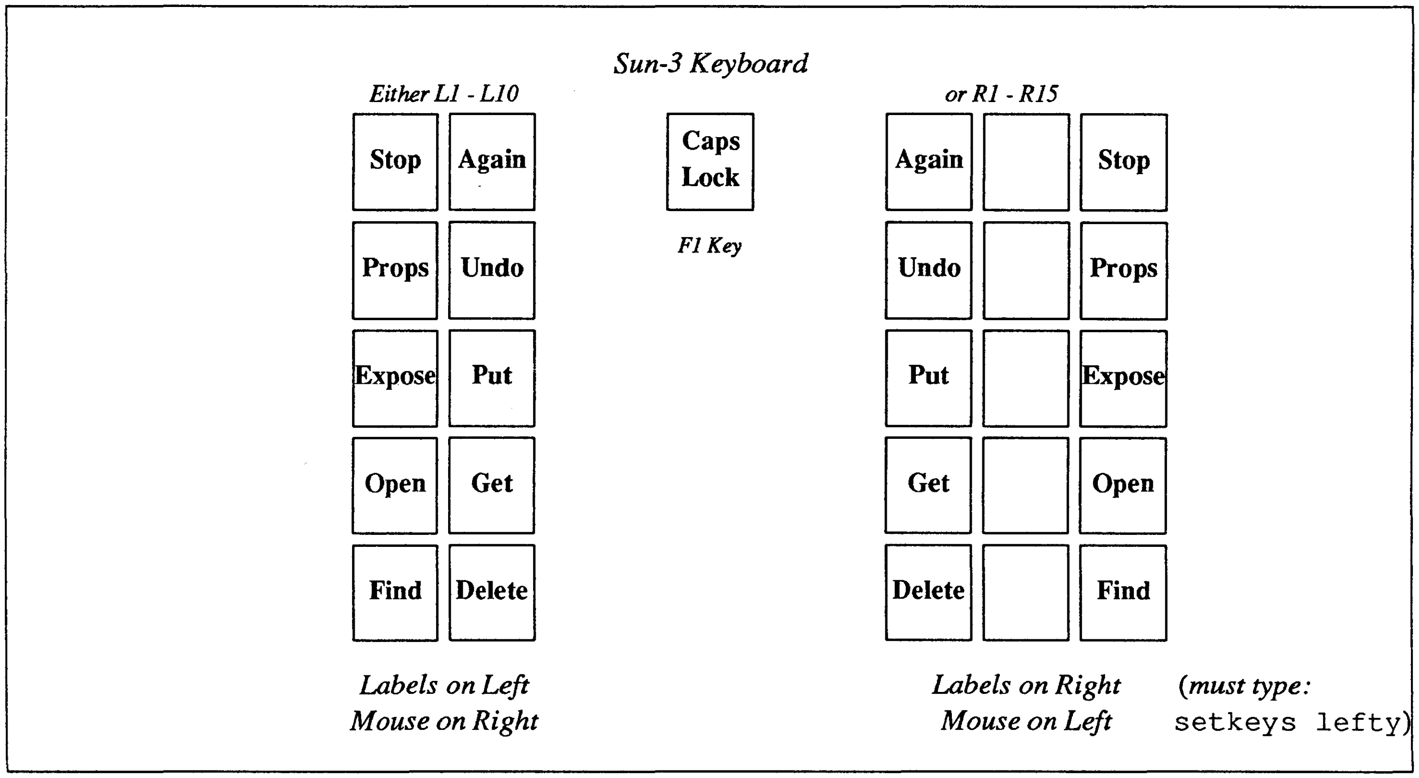
Basic Editing

Select	$\boxed{\text{LEFT}}$ to select, $\boxed{\text{MIDDLE}}$ to adjust.
Scroll	$\boxed{\text{LEFT}}$ in scroll button to scroll forward one line, $\boxed{\text{RIGHT}}$ to scroll backward, $\boxed{\text{MIDDLE}}$ to scroll a page.
Insert	Type characters.
Delete	$\boxed{\text{Delete}}$ function key. With characters adjacent to the caret, you can use UNIX rubout, erase word, or line kill characters.
Reinsert deleted text	$\boxed{\text{Get}}$ or $\boxed{\text{CTRL-G}}$.
Replace	$\boxed{\text{Delete}}$, type characters.
Move	$\boxed{\text{Delete}}$, select destination, $\boxed{\text{Get}}$.
Store on shelf	$\boxed{\text{Put}}$.
Copy from shelf	$\boxed{\text{Get}}$ or $\boxed{\text{CTRL-G}}$.
Copy text from elsewhere	Select text, $\boxed{\text{Put}}$, select destination, $\boxed{\text{Get}}$.
Undo edits	$\boxed{\text{Undo}}$ undoes all edits since last selection, or use of $\boxed{\text{Get}}$, $\boxed{\text{Put}}$, or $\boxed{\text{Delete}}$.
Load a file	`Load file` item on text menu, or type name of file to empty text subwindow, followed by $\boxed{\text{ESC}}$. `Reset` text menu item (once or twice) makes window empty.
Save	`Save` or `Store to named file` item on text menu.
Exit	`Quit` item on frame menu.

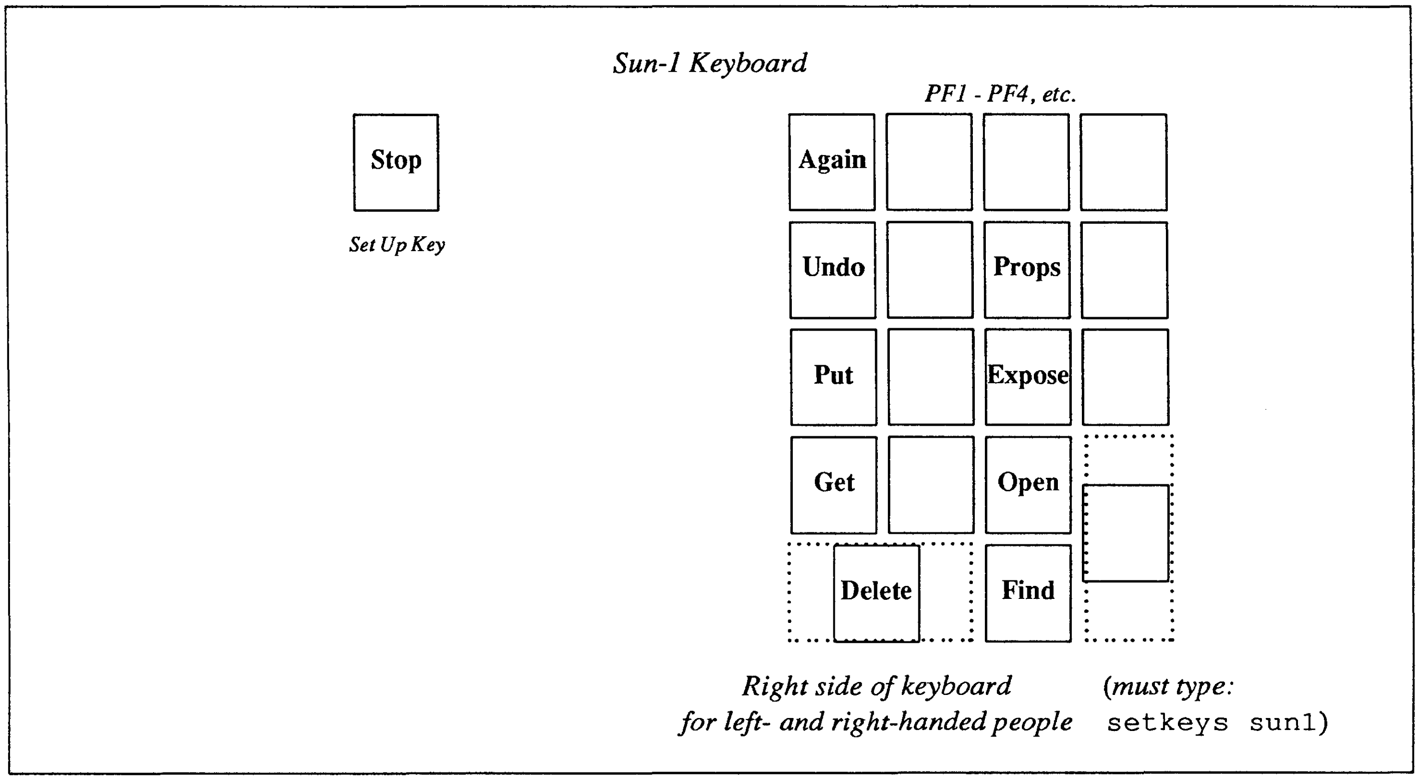
Function Keys



Function Keys *cont.*



(*must type:*
`setkeys lefty`)



(*must type:*
`setkeys sun1`)

Corporate Headquarters
Sun Microsystems, Inc.
2550 Garcia Avenue
Mountain View, CA 94043
415 960-1300
TLX 287815

For U.S. Sales Office locations, call:
800 821-4643
In CA: 800 821-4642

European Headquarters
Sun Microsystems Europe, Inc.
Berkshire House
High Street
Ascot, Berkshire SL5 7HU
England 0990 28911
TLX 846573

Germany: 089 416 00820
UK: 0990 25942
France: 01 630 23 24

Canadian Headquarters
416 477 6745

Europe, Middle East, and Africa, call European Headquarters:
0990 28911

Elsewhere in the world, call Corporate Headquarters:
415 960 1300
Intercontinental Sales