Facet*Term* ®

Version 3

Multiple Session Manager
for Standard Character Terminals

FacetCorp

Trademarks and Copyright

Facet***Term***® is a registered trademark of Facet Corp.

UNIX is a registered trademark of The Open Group.

All other product names are trademarks or registered trademarks of their respective companies.

Facet***Corp*** 1820 Preston Park Blvd., Suite 1000 Plano, TX 75093 USA
Telephone: 1-877-322-3846 (1-877-FacetGo) or 1-972-985-9901
FAX: 1-800-982-9901 or 1-972-612-2035

READ THIS FIRST!

You Must Register Your Software or it will stop running after 30 days.

All of the Facet*Term* software we ship is "evaluation" software in the sense that it will stop running after 30 days unless it is provided with an authorization key. This allows us to let all of our customers try the fully functional product before they buy it.

Here's how the process works:

□ After you have evaluated Facet*Term* and decide to buy a license, you are given an “Authorized Facet*Term* Registration” form which we call an “AFR”.

□ You should follow the instructions on the form for filling it out and faxing it to us.

□ In response to your AFR fax, we will fill in the space for your registration number on the form and fax it back to you.

□ The return fax will include instructions for entering the registration number on your system. You should then file your AFR in a safe place.

We will track your license by the ***AFR tracking number*** at the bottom of the AFR form. ***Write this tracking number in the space provided below*** so that you will always be able to easily refer to this number.

AFR Tracking Number

You can get started quickly by only reading a small part of the manual.

We realize that most people like to start using a new software package as quickly as possible without reading the manual. We’ve tried to design the manual such that you can do a minimum amount of reading before beginning to use Facet*Term*.

The manual begins with an introduction which briefly describes what Facet*Term* is and what benefits it provides. This is followed by a very short tutorial which teaches you to use the main features of Facet*Term* quickly. While reading the tutorial, run Facet*Term* and perform each step as you read about it in the tutorial. The tutorial will not introduce you to all of Facet*Term*'s features, but it will get you started with the main feature -- running multiple sessions on your terminal.

Later, as you want to learn more about Facet*Term*, you can refer to the Users Guide chapter, Configuration Guide chapter and Window Command Line Reference chapter for detailed information about each Facet*Term* feature and option. You will also find a file named "README" installed in the /usr/facetterm directory on your machine. This file will contain up-to-date information not in the manual, and may refer you to other more specific "README" files.

License Agreement / Warranty

License Agreement

This Software is the property of Facet Corp. and is protected by both United States Copyright Law and International Treaty provisions. You are granted a license to use this Software under the terms stated in this agreement. You may move the Software from one computer to another, but at no time may a single copy of the Software be installed on more than one computer. Facet Corp. authorizes you to make archival copies of the Software for the sole purpose of backing up your software and protecting your investment from loss. Any other use or transfer of the Software without written permission is in violation of Facet Corp's Copyright.

Title/Ownership

Facet Corp. retains title, copyrights, intellectual property rights, and ownership in the Software. This is not a sale of the Software. You are purchasing only the media upon which the Software is recorded.

Limited Warranty

With respect to the physical diskette or tape and physical documentation enclosed herein, Facet Corp. warrants the same to be free of defects in materials and workmanship for a period of thirty (30) days from the date of purchase by the end-user customer. Licensee may return any defective media to their supplier during the warranty period for a replacement free of charge. The remedy for breach of this warranty shall be limited to replacement and shall not encompass any other damages, including but not limited to loss of profits, special, incidental, consequential, or other similar claims.

Facet Corp. further warrants that the Product(s) work substantially as described in the documentation, when properly installed, configured and used, for a period of thirty (30) days from the date of purchase by the end-user customer. Facet Corp. may resolve material Product(s) defects at Facet Corp's option by any one of: 1) correct the software defect; 2) modify the documentation to correspond to the Product(s) performance; 3) replace the Product(s); or 4) refund the purchase price of the Product(s). Facet Corp. does not warrant that the Product(s) will meet the customer's specific requirements.

FACET CORP. SPECIFICALLY DISCLAIMS ALL OTHER WARRANTIES, EXPRESSED OR IMPLIED, INCLUDING BUT NOT LIMITED TO, IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE. IN NO EVENT SHALL SSSI BE LIABLE FOR ANY LOSS OF PROFIT OR ANY OTHER COMMERCIAL DAMAGE, INCLUDING BUT NOT LIMITED TO SPECIAL, INCIDENTAL, CONSEQUENTIAL, OR OTHER DAMAGES.

This agreement shall be governed by the laws of the State of Texas.

Acknowledgment

You acknowledge that you have read this license agreement and limited warranty and agree to be bound by its terms and conditions. You agree that this agreement supersedes any and all prior oral and written communications relating to the subject matter hereof.

Support / Maintenance

Support Policy

Facet***Corp*** will provide support to users of this software for a period of 30 days from the date of purchase.

Maintenance

Support may be extended beyond the initial thirty day warranty period by purchasing extended maintenance coverage. Extended maintenance includes:

❑ Software support for Facet*Term* via telephone, fax or electronic mail.

❑ Upgrades to any new versions of Facet*Term* for a nominal media and shipping charge. Facet*Term* upgrades are beneficial, not only to receive new Facet*Term* features, but are often required to support new operating system releases, new terminals, or new platforms.

For more information on purchasing Facet*Term* Maintenance, contact your distributor.

Contents

Introduction

This chapter gives you a brief overview of Facet*Term.* The following topics will be discussed:

❑ What Facet*Term* is and what it can do for you

❑ Installing Facet*Term*

❑ Your UNIX environment

❑ What to do when you need technical support

❑ How to contact us at Facet***Corp***.

What is Facet*Term*?

FacetTerm lets you run up to 10 sessions on a single terminal.

Facet*Term* makes a standard “dumb” character terminal appear to be 10 “terminals”. Each of the 10 Facet*Term* windows can run its own independent program. You can quickly switch from one window to another at any time. On a standard character terminal, Facet*Term* windows are primarily used as full screen windows.

Benefits

Facet*Term* can increase your productivity by allowing you to have several applications active at once. For example, you can be in the middle of editing a document with a word processor in one window, then jump to another window to look up information in a database. When you return to the word processing window, it is exactly as you left it. Facet*Term* saves you the trouble of having to exit one application to use another.

Features

Facet*Term* provides the following features on a standard character terminal:

❑ Multiple terminal sessions

❑ Copy and paste of information between sessions

❑ Printing of the contents of a session window

❑ Support of a local printer attached to your terminal

❑ Notification capabilities, like watching for new mail, or watching for a process to complete in an off-screen window

❑ A pull-down menu user interface which is customizable

❑ A command line user interface

Installing Facet*Term*

The installation procedure for Facet*Term* varies depending on the type of UNIX system you have. You should follow the installation instructions that were shipped with the Facet*Term* software for your particular system.

Your UNIX Environment

The example sessions presented in the next chapters assume that you are starting by running a UNIX shell on the terminal where you want to run Facet*Term.* If your login account is set up to automatically run something other than a shell when you login, then you should use another account for the purposes of the tutorial. Later you will learn how to set up your account so that when you login, Facet*Term* is automatically started, and your applications are automatically started in Facet*Term* windows.

Be sure that the UNIX environment variable $TERM is set to the appropriate terminal type.

Getting Technical Support

Quite often your dealer or distributor will want to be "in the loop" when you need technical help with Facet*Term.* They may be familiar with your specific computing environment, and therefore able to furnish information about the problem that we would not otherwise know about. If you are outside North America, time zone or language differences may make it more convenient to work with your local vendor. Therefore, when you need help, please check first with your dealer or distributor to see if they can support you.

However, please know that we want your problems to be solved, and have a capable and willing staff of support engineers ready to help you resolve any problems that you may have with Facet*Term.*

The following information is usually required to help you with a problem. Please try to have it available when you call for help.

1. ***The FacetTerm part number.*** This can be found on the diskette or tape label on your Facet*Term* installation media. In addition, if you have Facet*Term* installed on your system you may get this information by running the Facet*Term*

administration menu:

```
facetadm
```

Then choose the item called "Display version information". All version information about your copy of Facet*Term* will be displayed.

2. ***The setting of your TERM environment variable.*** This can be determined with the command:

   ```
   echo $TERM
   ```

   This information is also displayed by the facetadm program when you choose "Display version information".

3. ***The setting of the FACETTERM environment variable***, if set. This can be determined with the command:

   ```
   echo $FACETTERM
   ```

   This information is also displayed by the facetadm program when you choose "Display version information".

How to Contact Us

You may contact our Sales or Support staff at the following phone numbers, fax numbers, and E-mail addresses:

Phone:	(877)322-3846 (972)985-9901
Fax:	(800)982-9901 (972)612-2035
Internet:	info@facetcorp.com support@facetcorp.com

Sales and support are often handled best via fax or e-mail. We avoid telephone tag, and each have the opportunity to compose our questions and answers more carefully.

Quick Start

This chapter leads you through an initial tutorial session which will familiarize you with Facet*Term*'s fundamental features.

In this chapter, you will learn:

❑ How to start Facet*Term*

❑ How to use the Facet*Term* menu interface

❑ How to use the Facet*Term* command line interface

❑ How to switch windows

❑ How to run programs in windows

❑ How to quit Facet*Term*

Tutorial

The quickest way to learn Facet*Term* is to start using it as soon as possible. This section will lead you through a session using the most common features of Facet*Term*.   You are strongly encouraged to follow this tutorial session on your terminal. We have intentionally made it very brief. Taking the few minutes required to follow this tutorial will allow most users to learn what they need to know to start using Facet*Term* right away.

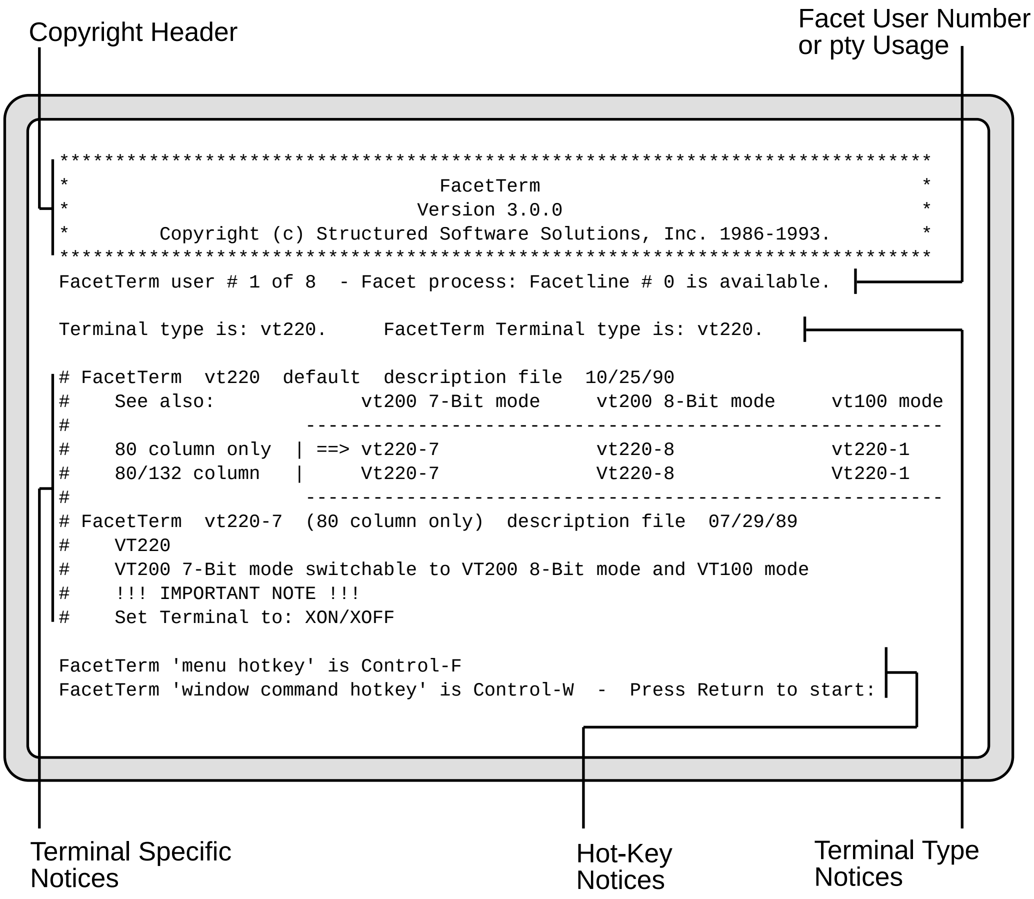
Starting Facet*Term*

Start FacetTerm by running the "facetterm" command.

To start Facet*Term* on a standard terminal, at the UNIX shell prompt type:

facetterm RETURN

As Facet*Term* starts you will see the start-up screen which will give you some information which you should verify:



```
******************************************************************************
*                                FacetTerm                                   *
*                              Version 3.0.0                                 *
*        Copyright (c) Structured Software Solutions, Inc. 1986-1993.        *
******************************************************************************
FacetTerm user # 1 of 8  - Facet process: Facetline # 0 is available.

Terminal type is: vt220.     FacetTerm Terminal type is: vt220.

# FacetTerm  vt220  default  description file  10/25/90
#    See also:            vt200 7-Bit mode     vt200 8-Bit mode     vt100 mode
#                     --------------------------------------------------------
#    80 column only  | ==> vt220-7              vt220-8              vt220-1
#    80/132 column   |     Vt220-7              Vt220-8              Vt220-1
#                     --------------------------------------------------------
# FacetTerm  vt220-7  (80 column only)  description file  07/29/89
#    VT220
#    VT200 7-Bit mode switchable to VT200 8-Bit mode and VT100 mode
#    !!! IMPORTANT NOTE !!!
#    Set Terminal to: XON/XOFF

FacetTerm 'menu hotkey' is Control-F
FacetTerm 'window command hotkey' is Control-W  -  Press Return to start:
```

Review the start-up screen, looking for notices about terminal setup and the FacetTerm hot keys.

Facet*Term* will report the type of your terminal as specified by the value of your $TERM environment variable. For now, the Facet*Term* terminal type will be the same as the $TERM terminal type. The User Guide Chapter, later in this manual, explains how to change the Facet*Term* terminal type to be more specific. This is sometimes necessary to support special features of your terminal.

The Facet*Term* terminal description files contain information which will be printed on the start-up screen. These notices give you instructions on how the terminal should be set up, indicate what limitations the description might have in the way that it runs your terminal, and often refer you to other descriptions.

At the bottom of the start-up screen, Facet*Term* shows the currently defined hot-keys for accessing the menu interface and the window command line interface.

If the information on the start-up screen doesn't match your terminal setup, press the interrupt key. Otherwise, press the return key to continue.

If you think any of the information shown on the start-up screen does not match the way your terminal is set up, you should interrupt Facet*Term* by pressing the interrupt key. This key is usually defined as:

DEL

Otherwise, if everything looks correct, continue by pressing:

RETURN

Facet*Term* will now start running programs in windows as specified in the .facet file.

The .facet File

The FacetTerm start-up configuration file is called ".facet". The default .facet file will start shells in windows 1 and 2 and the pull down menu in window 10.

When Facet*Term* starts, it reads a start-up configuration from a file named ".facet". The default .facet file is in the /usr/facetterm directory. It specifies that Facet*Term* is to start shells in windows 1 and 2 and the pull-down menu in window 10. You may copy this file to your home directory and customize it to your own needs. This file has many comments in it which explain your options in customizing your Facet*Term* start-up environment. A comment in the .facet file is a line beginning with a "#". In most cases, you can turn on a feature by simply taking out the "#" and leading spaces in a line.

The shell prompt that you have on your screen now is from the shell running on window 1.

You should now have a shell prompt on your screen. You are interacting with a shell which is running on Facet*Term* window 1. You can verify that you are running a shell on a window by running the UNIX "tty" command:

tty RETURN

You will notice that the shell is running on a device different from your normal terminal device. Each Facet*Term* window appears to be an independent login session to UNIX. Verify this by running the UNIX "who" command:

who RETURN

You will notice that you are shown to be logged in four times; once on your normal tty device, twice on the windows where the shells are running, and once on the window where the menu is running.

As long as you do not press either of the Facet*Term* hot-keys, all of your keystrokes will be directed to the program running in the current window, and the output from that program will be put on the screen as normal.

Facet*Term* does not make any changes to the appearance of your screen as you work with an application. It is only when you press a hot-key that you begin interacting with Facet*Term*.

Using the Pull-Down Menu Interface

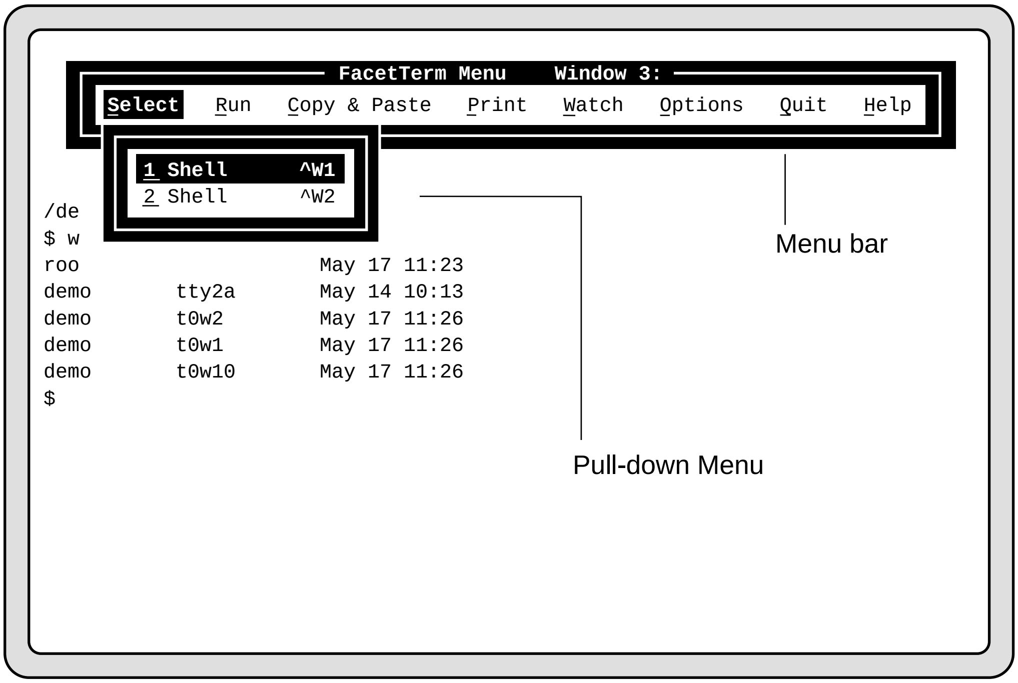
Press the menu hot-key (Ctrl-F) and the FacetTerm menu will pop up over your current window.

The menu interface is accessed by the menu hot-key. The default menu hot-key is Ctrl-F:

CTRL F

(Hold down the Ctrl key while you press the F key - think of “F” for Facet*Term*).

Go ahead and press the menu hot-key now. The menu should pop up over your current window:



```
/de
$ w
roo                    May 17 11:23
demo       tty2a       May 14 10:13
demo       t0w2        May 17 11:26
demo       t0w1        May 17 11:26
demo       t0w10       May 17 11:26
$
```

The menu consists of a menu bar which contains selections for major function categories. When an item on the menu bar is selected, a pull-down menu will drop down, displaying further menu choices. The menu has come configured to automatically pull down the window selection menu when it is accessed.

If one of your applications uses Ctrl-F, you can get the key to your application by pressing it twice. The first one will cause the menu to pop up and the second one will cancel the menu and pass on through to the application running in the current window.

Use the arrow keys to move along the menu bar and up and down a pull-down menu. Select the highlighted item by pressing Return. Cancel a menu by pressing the space bar or escape key.

Navigating the menu is easily done with a few keys. To move left or right on the menu bar use the left and right arrow keys:

← and →

If a pull-down menu is selected, moving across the menu bar will cause the next pull-down menu to be selected. If no pull-down menu is selected, moving across the menu bar will simply move the highlighted selection bar (this is a quicker way to move across the menu bar if you do not need to see all the pull-down menus). Once the selection bar is highlighting the item you would like to select, simply press:

RETURN

to select it. Or, rather than moving the selection bar, you can directly select a menu item by pressing the key whose letter (or number) is underlined (or otherwise highlighted) in the item name.

The selection bar on a pull-down menu is moved from one item to another by using the up arrow and down arrow keys:

↑ and ↓

To cancel a menu press the space bar or escape key:

␣ or ESC

(using the space bar is faster).

When you cancel the menu bar, the menu is completely removed and you are once again interacting with the program on the current window. Go ahead and press the space bar until the menu bar is removed.

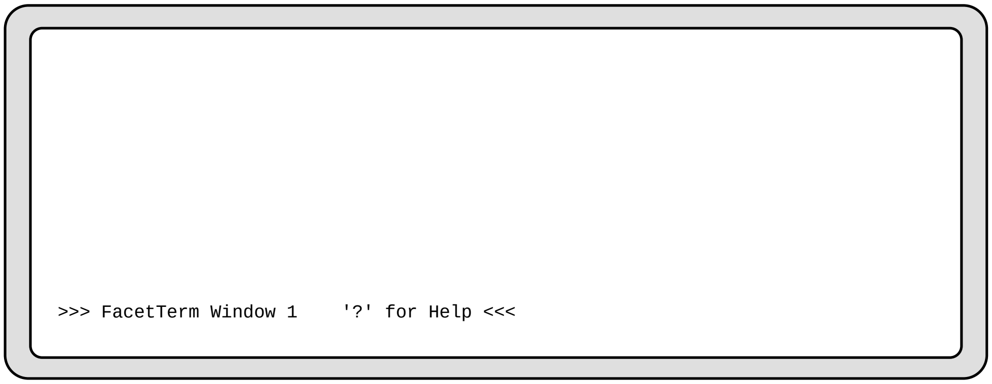
Using the Window Command Line Interface

The Facet*Term* window command line interface provides a quick method of accessing all of Facet*Term*'s features. To bring up the Facet*Term* window command line press Ctrl-W:

Press the command line hot-key (Ctrl-W) and the FacetTerm command line will come up at the bottom of the screen.

CTRL W

(Hold down the Ctrl key while you press the W key - think of "W" for Window). Press the window command line hot-key now. The Facet*Term* window command line will appear at the bottom of your screen:



```
>>> FacetTerm Window 1    '?' for Help <<<
```

The command line will always tell you which window you are on and remind you that you can get help by pressing:

?

If one of your applications uses Ctrl-W, you can get the key to the application by pressing it twice.

The FacetTerm command line accepts single character commands. Cancel the command line with the space bar or escape key.

When the window command line is on the screen, your keyboard input is directed to Facet*Term* rather than the current window. The window command line allows you to direct the actions of Facet*Term* with single key commands. Some commands lead to further options or require the input of a name. In either of these cases, the command line will prompt you for this additional input.

Most commands will cause the window command line to be removed, and leave you connected to the current window. However, some commands leave the command line active for convenience. When the window command line is active, pressing

RETURN

before any other key command is entered will cause Facet*Term* to refresh the current window and remove the command line. You may exit the window command line without performing any function by pressing:

␣ or ESC

Go ahead and remove the window command line now.

Selecting Windows with the Menu Interface

Bring up the menu by pressing:

CTRL F

Notice that the Select item has been automatically selected, and the window selection menu is pulled down. The Facet*Term* menu is installed this way by default because window selection (or simply looking at the select menu to see what is running where) is probably the most common use of the menu.

Select a window using the menu, by choosing from the "select" menu.

The window selection menu shows that shells are running in windows 1 and 2. The shell running in window 1 is highlighted because it is your current window. As explained above, you can move the highlight from one item to another in a pull-down menu by using the up and down arrow keys. You can also directly select a window by pressing the window number. At this point, you should move the highlight to the item for the shell running in window 2 and select it by pressing

RETURN

or press

2

to select window 2. The menu will be removed, and Facet*Term* will switch to window 2.

Selecting Windows with the Command Line Interface

To switch windows with the Facet*Term* command line, access the command line as usual with:

CTRL W

Select a window using the command line interface by pressing Ctrl-W and the number of the window you want to switch to.

Once the Facet*Term* command line is visible, to switch back to Window 1, press:

1

You should see the screen refreshed to be as it was before you switched away from window 1. After you have switched windows, the command line is taken away and you are once again interacting with the program in the selected window. Therefore, to switch back to window 2, you must bring up the command line again with

CTRL W

and then select window 2 with:

2

Running Programs with the Menu Interface

You can easily start additional programs in new windows with either the menu or command line. To start a new program with the menu, pop up the menu with the menu hot-key:

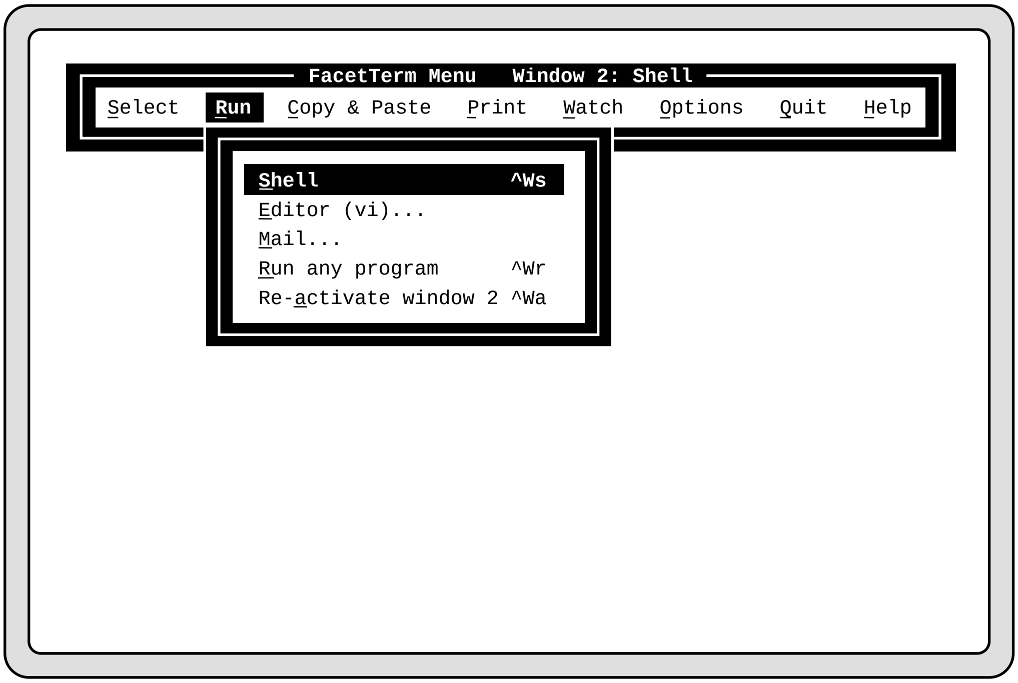
CTRL F

Start new programs using the menu by going to the "Run" menu and choosing the program you want to start in an idle window.

The menu bar will pop up and the window selection menu will be automatically selected. Move to the “Run” menu by pressing the right arrow key:

→

This will cause the “Select” menu to be removed, and the “Run” menu will be pulled down:



The “Shell” item will be highlighted since it is the first item in the menu. To start a shell in a new window press:

RETURN

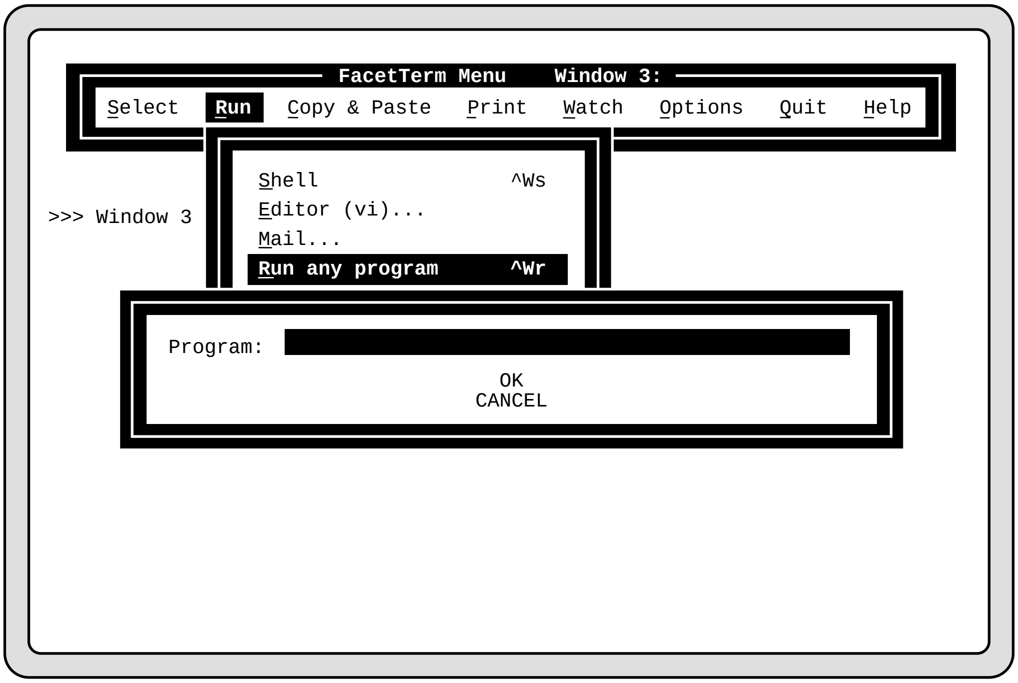
The menu will be removed, the next available window will be selected, and the new shell will be run there. In this case, the shell will be run on window 3.

When a window goes idle, the menu will pop up to allow you to select another window or start another program.

Now, let’s exit this shell and run something else in its place. To exit the shell, at the shell prompt type:

exit RETURN

Notice that when a window goes idle, the menu pops up to allow you to select an active window or start a new program. Once again move right to the “Run” menu. This time, select the “Run any program” item on the menu. This will cause a dialog box to be presented:



```
>>> Window 3
```

If the program you want to start isn't listed on the "Run" menu, you can start it by choosing the "Run any program" item and supplying the command to run the program.

The dialog box is prompting for the name of a program to be run. You should give it the name of the program just as you would if you were running the program from a shell. For this example let’s run the UNIX “cal” program. This program will display a calendar and then exit. After you enter the name of the program to run, the selection bar will be positioned on the “OK” item in the dialog box. If the information you supplied in the dialog box is correct, you should select the “OK” item by pressing

RETURN

If the information is not correct, you may reposition to the item prompting for the information and re-enter the information. You may cancel a dialog box (and the operation that was requesting the information) by positioning to the “Cancel” item and selecting it. If you have entered the “cal” program name correctly, go ahead and select the “OK” item. This will cause the calendar program to be executed in window 3.

Because the window goes idle when the cal program finishes, the menu will pop up again. Select window 1 from the window selection menu. Next you will learn how to run programs with the command line interface.

Running Programs with the Command Line Interface

To run a shell on an idle window using the command line interface, bring up the command line with:

CTRL W

and then press:

To start a shell using the command line, press Ctrl-W and then S.

S

(for Shell). This will cause the next available window (window 3 in this case) to be selected and a new shell will be started there. You will now be interacting with the shell on window 3.

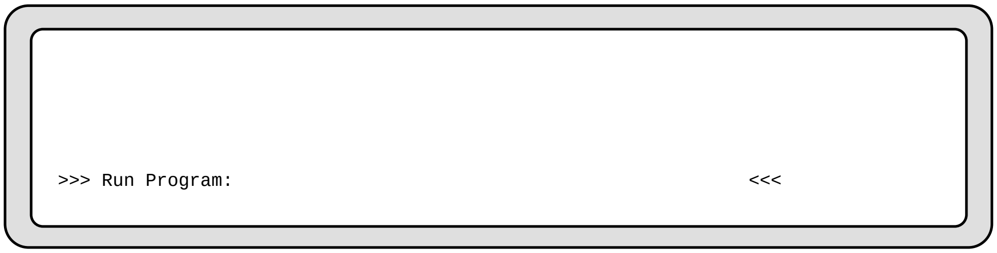
To run a program other than a shell on an idle window press:

CTRL W

and then

R

Facet*Term* will prompt:



```
>>> Run Program:                                           <<<
```

To run any program, using the command line, press Ctrl-W and R, and then the command for the program you want to run.

Type the name of the program you want to run and the return key. If you type:

vi RETURN

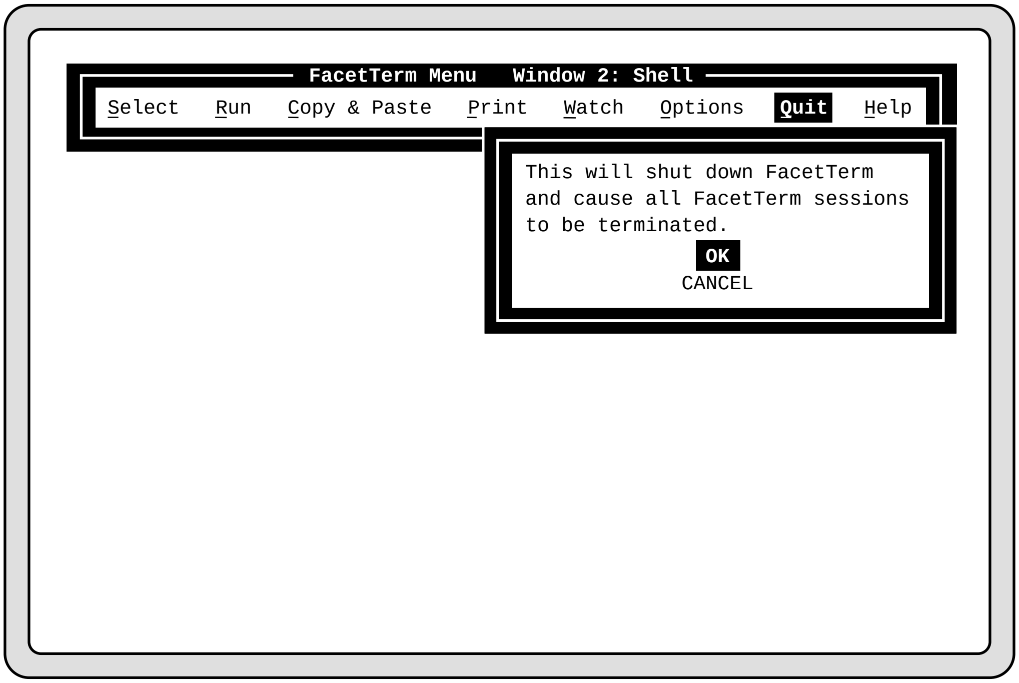
the UNIX “vi” editor will be run in the next available window.

Quitting Facet*Term* with the Pull-Down Menu

To quit Facet*Term* you may pop up the menu and select the “Quit” item on the menu bar. A quick way to do that is:

CTRL F ␣ Q

A dialog box will be presented with a warning that the programs running in all active windows will be terminated:


FacetTerm Menu Window 2: Shell
Select Run Copy & Paste Print Watch Options Quit Help

```
This will shut down FacetTerm
and cause all FacetTerm sessions
to be terminated.
```

If you want to quit, select the “OK” item in the dialog box. For now, move down to the cancel item, and press return to cancel the quit request.

Quitting Facet*Term* with the Command Line

To quit FacetTerm using the command line, press Ctrl-W and then Q. You must then confirm that you want to quit by pressing Y.

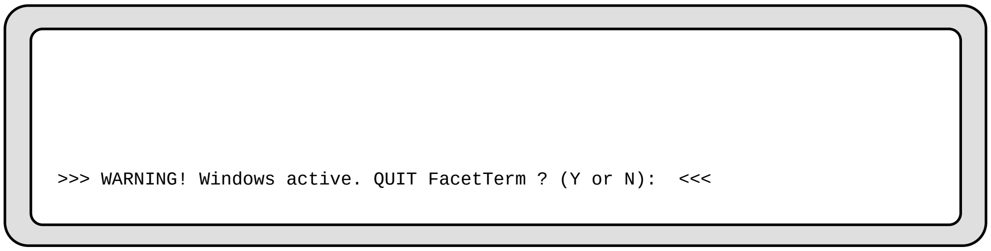
To quit Facet*Term* from the command line, bring up the command line with

CTRL W

and then press

Q

Because you have not exited the programs in all your windows, you will see the message warning that there are still windows which are active:



```
>>> WARNING! Windows active. QUIT FacetTerm ? (Y or N):  <<<
```

If you answer:

Y

any programs running on active windows will be terminated and then Facet*Term* will terminate, leaving you interacting with the shell that was running on your terminal before you started Facet*Term*.

If you answer:

N

you will still be running Facet*Term* and will return to your current window. Go ahead and answer “Yes” to the question, and Facet*Term* will quit.

Warnings

In general, running your applications in Facet*Term* windows will be no different than running them directly on your terminal. One exception to this is the fact that you may quit Facet*Term* and cause it to terminate your applications running in windows with a software signal. This is probably not the way that you usually terminate your applications. **You should be sure to save any changes which you have made to data with your applications before quitting Facet*Term* and having it terminate them. Better yet, simply exit all applications running in windows before quitting Facet*Term*.**

You should **NEVER** terminate the "facetterm" process with the command:

```
kill -9
```

This does not give Facet*Term* the opportunity to terminate processes running on windows, and will therefore leave these processes running. The Facet*Term* devices that these processes are running on will be unusable until the processes are killed manually.

Facet*Term* Menu Summary

Remember, you can access the Facet*Term* menu with:

CTRL F

Move among items on the menu bar with the left and right arrow keys:

← and →

Move among items in a pull-down menu or dialog box with the up and down arrow keys:

↑ and ↓

Select an item with:

RETURN

cancel a menu with:

␣ or ESC

Window Command Line Summary

Remember, you can access the Window command line with:

CTRL W

After the command line is accessed, select windows with:

1 through 9 and 0 for window 10

Run a shell on a new window with:

S

Run any other program on a new window with:

R

Quit Facet*Term* with:

Q

Refresh the current window with:

RETURN

Cancel the command line with:

□ or ESC

Facet*Term* User's Guide

This chapter explains in detail each of the "user" features of Facet*Term.*

In this chapter, you will learn:

- ❏ All the options you can use when starting Facet*Term*
- ❏ What to look for on the start-up screen
- ❏ How to start Facet*Term* from your .profile
- ❏ How to specify what programs you want started in windows automatically when you start Facet*Term*
- ❏ How to use each of the Facet*Term* user interfaces
- ❏ How to select windows
- ❏ How to start additional programs after Facet*Term* is running
- ❏ How to use copy and paste
- ❏ How to print the screen
- ❏ How to use the Window Watch feature
- ❏ How to create simple key macros
- ❏ How to program function keys with Facet*Term*
- ❏ How to use the screen saver and screen lock features
- ❏ How to quit Facet*Term*

Starting Facet*Term*

To run Facet*Term*, log in as you normally do, and at the shell prompt type:

```
facetterm options RETURN
```

where *options* is a list including any of the options described below.

If your login runs a menu or application automatically, you will probably want to modify your profile to have it run Facet*Term* instead. Please refer to the section below titled “Facet*Term* and .profiles”.

Another way to start Facet*Term*, so that you log off automatically when you quit Facet*Term*, is to use the exec command:

```
exec facetterm options RETURN
```

Shells that are started on Facet*Term* windows will use the .profile in the directory where you start Facet*Term*. This means that you will normally want to start Facet*Term* from your home directory. There may be some commands in your .profile which should be executed when you first login, but which are unnecessary or improper to have executed on each window. This will be discussed in the section titled “Facet*Term* and .profiles” later in this Chapter.

Facet*Term* Start-up Options

You may specify the following options when starting Facet*Term*.

Using an Alternate Terminal Description

You may determine that you want to use an alternate Facet*Term* terminal description. For example, if you set your $TERM variable to “wyse50”, but you have a Wyse 60, then you will want to have Facet*Term* take advantage of the screen pages that the Wyse 60 has. You would, therefore, start Facet*Term* with an alternate terminal description:

```
facetterm  wy60wy50   RETURN
```

An alternate terminal description can also be specified with an environment variable (see the section titled "Environment Variables Used by Facet*Term*" in the Configuration Guide chapter).

Your dealer or a Facet***Corp*** support engineer can advise you on the use of appropriate terminal descriptions.

Starting Facet*Term* in Nonstop Mode

Facet*Term* will skip the start-up notices and the “Press Return to Start” prompt if "nonstop" is added to the start-up command:

```
facetterm  nonstop   RETURN
```

You should only start Facet*Term* this way after you are comfortable with your day to day Facet*Term* operating environment.

Limiting the Number of Windows

You can limit the number of windows that Facet*Term* uses, thereby reducing system memory requirements. This is done by including the maximum number of windows to be used on the Facet*Term* start-up command. For example,

```
facetterm  4    RETURN
```

will limit the number of windows to 4. If you do not limit the number of windows, Facet*Term* will use enough memory to run 10 windows.

Please note that the .facet file comes configured to run the menu interface program in window 10. Therefore, if you limit the number of windows, you must be sure to allow one additional window for the menu and modify the .facet file to indicate the proper menu window..

Changing the Window Command Line Hot-Key

The window command line hot-key can be changed or disabled by adding the "hotkey" specification to the start-up command. For example,

```
facetterm  hotkey=’^X’   RETURN
```

will change the hot-key to be

CTRL X

You should use the "**^**" character to indicate a control character, and you should enclose the key specification in quotes as shown. The hot-key can only be a single character.   If a series of characters is specified, only the first character will be used.

An alternate hot-key can also be specified with an environment variable (see the section titled “Environment Variables Used by Facet*Term*” in the Configuration Guide chapter), or in the .facet file (see the section titled “.facet File Reference” in the Configuration Guide chapter).

To disable the hot-key, use the "hotkey" option without a key specification:

```
facetterm  hotkey=   RETURN
```

Note that disabling the hot-key should only be done when the menu is being used. Otherwise, the user will not be able to quit Facet*Term*.

Changing the Menu Hot-Key

The menu hot-key can also be changed or disabled by adding the "menu_hotkey" specification to the start-up command. For example,

```
facetterm  menu_hotkey='^G'   RETURN
```

will change the menu hot-key to be

CTRL G

You should use the "**^**" character to indicate a control character, and you should enclose the key specification in quotes as shown. The menu hot-key can only be a single character. If a series of characters is specified, only the first character will be used.

Keys that You Should Not Use as Hot-Keys

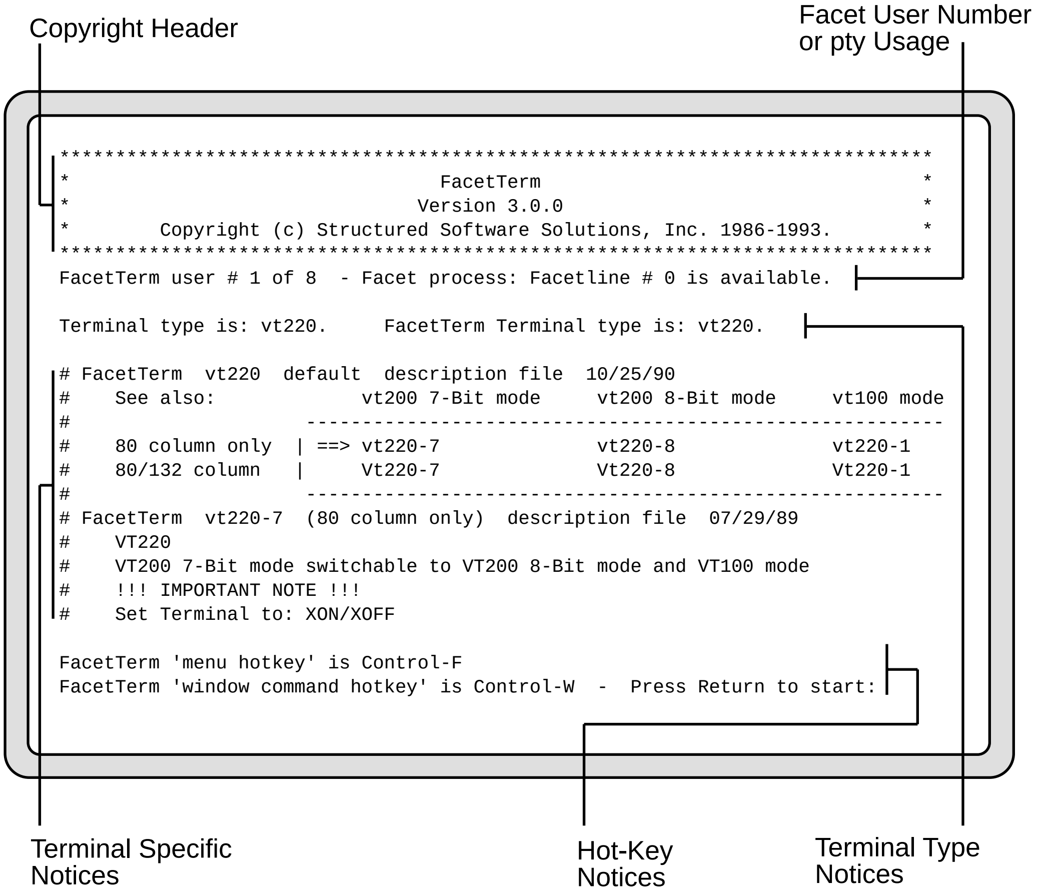
The following are some keys that you should not use as Facet*Term* hot-keys:

Ctrl-M	carriage return character
Ctrl-S	Xoff character
Ctrl-Q	Xon character

There may be other keys on your system which should not be used. Check with your system administrator if you're not sure.

Facet*Term* Start-up Screen

When Facet*Term* starts, it presents the following start-up screen (unless it has been started in non-stop mode):



```
******************************************************************************
*                                 FacetTerm                                  *
*                               Version 3.0.0                                *
*        Copyright (c) Structured Software Solutions, Inc. 1986-1993.        *
******************************************************************************
FacetTerm user # 1 of 8  - Facet process: Facetline # 0 is available.

Terminal type is: vt220.     FacetTerm Terminal type is: vt220.

# FacetTerm  vt220  default  description file  10/25/90
#    See also:             vt200 7-Bit mode     vt200 8-Bit mode     vt100 mode
#                     ---------------------------------------------------------
#    80 column only  | ==> vt220-7              vt220-8              vt220-1
#    80/132 column   |     Vt220-7              Vt220-8              Vt220-1
#                     ---------------------------------------------------------
# FacetTerm  vt220-7  (80 column only)  description file  07/29/89
#    VT220
#    VT200 7-Bit mode switchable to VT200 8-Bit mode and VT100 mode
#    !!! IMPORTANT NOTE !!!
#    Set Terminal to: XON/XOFF

FacetTerm 'menu hotkey' is Control-F
FacetTerm 'window command hotkey' is Control-W  -  Press Return to start:
```

Copyright Header and Registration Reminder

Facet*Term* will begin by displaying the version and copyright header. If Facet*Term* has not been registered, it will display a message reminding you of the number of days left before the evaluation period expires. If you allow this period to expire without registering Facet*Term*, it will no longer run. See the section titled “Read This First!” at the front of this manual.

Facet User Number or PTY Usage

This message will differ depending on the type of UNIX system you have. Facet*Term* installs its own pseudo-device driver on some systems and uses the standard UNIX pseudo-ttys (ptys) on other systems. If the Facet driver has been installed, the message indicating which Facet user you are will be displayed as shown in the example above. If the standard ptys are being used, this message will be replaced with the pty usage message indicating which ptys are being used for this instance of Facet*Term*. In either case, this message is for information only and does not affect your use of Facet*Term*.

Terminal Type Notice

The first terminal type is that specified by the UNIX $TERM environment variable. Facet*Term* uses this setting to select a terminal description file. This file tells Facet*Term* what control sequences to expect from applications and what functions the terminal can perform. The second terminal type will be the same, unless you specified an alternate terminal description file with the $FACETTERM environment variable or when you started Facet*Term*.

Special Terminal Notices

The Facet*Term* terminal description files contain information which will be printed on the start-up screen. These notices give you instructions on how the terminal should be set up, indicate what limitations the description might have in the way that it runs your terminal, and often refer you to other descriptions. This information will be printed below the terminal type notices.

Hot-Key Notice

At the bottom of the start-up screen, Facet*Term* shows the currently defined hot-keys for accessing the menu interface and the window command line interface. If any of the information on the start-up screen is incorrect, Facet*Term* can be safely interrupted at this point. Otherwise, pressing

RETURN

will cause Facet*Term* to start running the programs specified in the .facet file.

Facet*Term* and .profiles

If you wish to run Facet*Term* from your .profile or if you plan to have Facet*Term* run shells, there are a few issues you should be aware of.

If you simply put the "facetterm" command as the last line of your .profile, you will have the problem that any shells started on a window will also run the profile, and the command to start Facet*Term* will fail. There may also be other commands which should not be run by shells which are started on windows, such as "tset" which determines the terminal type based on the tty device name. The Facet*Term* package includes a utility program named fct_info which can be used in profiles to control execution of commands which should not be run if the shell is on a window. Your .profile should therefore have lines similar to the following at the end of the file:

```
if fct_info not_a_window
then
        # Put any command here which should not
        # run on a window
        tset
        # run FacetTerm with any options desired
        exec facetterm nonstop
fi
```

Terminal Description Files

Facet*Term* installs a collection of terminal description files. These files are installed in the /usr/facetterm/term directory. Each file describes to Facet*Term* how to use a particular terminal in a particular emulation or mode. These files are much more complex than standard “termcap” or “terminfo” files. This is because a Facet*Term* terminal description must contain all sequences which may be sent by all applications whereas a termcap simply specifies a single way to perform a subset of functions. In addition, the Facet*Term* terminal descriptions must specify many interactions between terminal features. For these reasons, the Facet*Term* terminal description files are *NOT* considered to be user configurable.

You might be asked by a Facet*Corp* support engineer to make specific modifications to a terminal description file as the resolution to a problem that you are having. If you do so, the file should first be copied to the /usr/facetterm/localterm directory and modified there. Facet*Term* will search for terminal descriptions there before searching in the /usr/facetterm/term directory.

Starting Programs Automatically with the .facet File

As Facet*Term* starts, it uses the .facet file to determine which programs should be started automatically. The .facet file is also used to control many other Facet*Term* features. These are described in various sections throughout this manual. A complete reference for the .facet file can be found in the "Configuration Guide" chapter.

When Facet*Term* is installed, a .facet file is installed in the /usr/facetterm directory. This default file instructs Facet*Term* to start shells in windows 1 and 2. You can customize the .facet file to your own needs. It is best to copy the default file to another directory and customize it there. Facet*Term* searches for a .facet file in the following order:

1. The directory in which Facet*Term* was started.

2. The user’s home directory as specified by the $HOME environment variable.

3. The /usr/facetterm directory.

4. The /usr/facet directory.

This allows the .facet file to be specified by application, by user, or for the whole system.

A .facet file consists of lines which specify programs to start automatically, or other options. Any blank line or line beginning with “#” is a comment. A line used to specify the automatic start-up of a program has the following form:

1. A window number in the first column. Use 0 for window 10, or L for the last available window.

2. A space or tab.

3. The name of the program that you would like to run in that window (it must be a single program - you can’t specify multiple programs separated with “;”). Any start-up parameters should also be given with the program name, just as you would run the program from a shell.

A window title line may be put before a program line:

window_title=*title*

This will specify the title for the window whose program line follows. So for example:

```
window_title=Shell
1 -sh

window_title=Word Processor
2 wp
```

will start a shell in window 1, run its .profile (the "-" before the sh causes the .profile to be run), and the window will be titled "Shell". A word processor started with the "wp" command will be run in window 2 and the window will be titled "Word Processor".

Facet*Term* User Interfaces

Facet*Term* provides two user interfaces. The ***window command line*** is the fastest way to access *all* of Facet*Term*'s features.

The ***FacetTerm pull-down menu*** provides an easy way to access the most common of Facet*Term*'s features. The Facet*Term* menu is primarily used on standard character terminals.

Each of these user interfaces is described in the following sections. Throughout the remainder of this chapter, as each Facet*Term* feature is described, access via the command line and the Facet*Term* menu will be discussed.

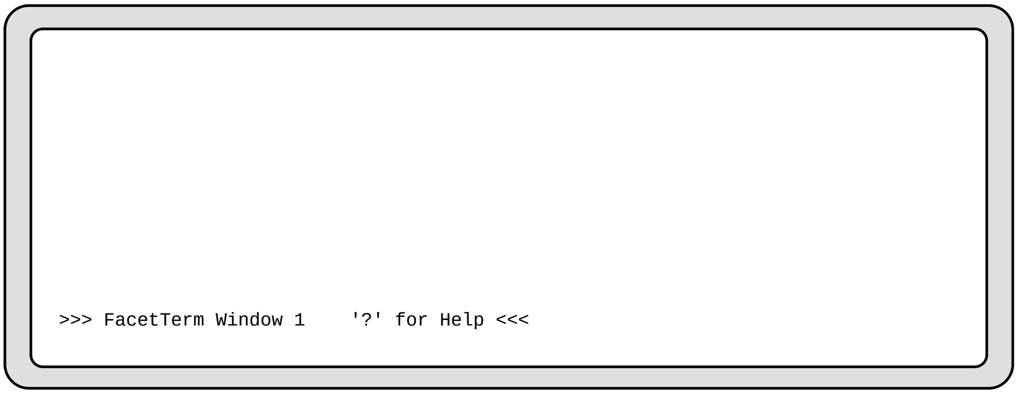
Using the Window Command Line

The Facet*Term* window command line is accessed by pressing the window command line hot-key. The default hot key is:

CTRL W

This hot-key can be redefined as an option when running the "facetterm" command. It can also be changed by redefining it in the .facet file. This method is described in the Configuration Guide chapter.

When you press the window command line hot-key, Facet*Term* will present the window command line on the last line of the screen:



```
>>> FacetTerm Window 1    '?' for Help <<<
```

The window command line accepts single character commands. Some of these commands lead to further command choices. Some commands leave the window command line active, ready for another command.   Others perform their function and then remove the window command line, leaving you interacting with the application in the current window.   The entire set of commands for the window command line is outlined in the Window Command Line Summary Chapter.

Using The Facet*Term* Menu

This section will describe in detail all of the elements of the Facet*Term* menu.

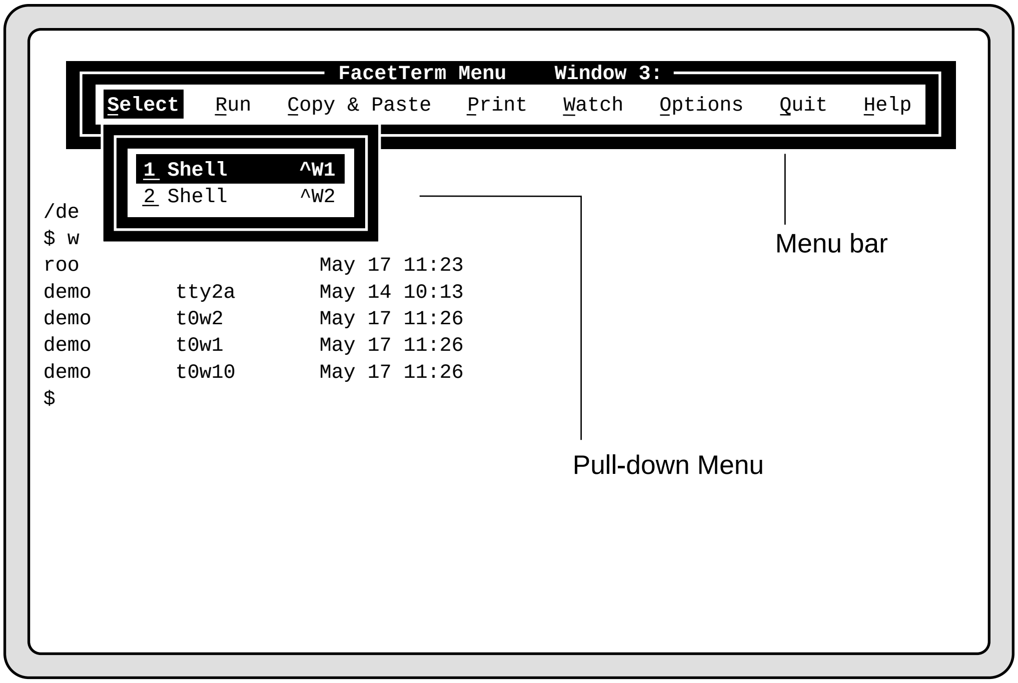
How the Menu Works

The menu is a Facet*Term* aware program which runs in a window. The default .facet file starts the menu in the last available window. The menu uses special calls to “pop up” over the current window when you press the menu hot-key. The default menu hot-key is

CTRL F

You should always access the menu with this hot-key rather than actually going to its window with a window selection command. The menu hot-key may be redefined either as an option to the "facetterm" command, or in the .facet file. All of the options of the .facet file are discussed in the Configuration Guide Chapter.

When you press the menu hot-key, it pops-up over the current window and automatically pulls down the window selection menu:



```
/de
$ w
roo                     May 17 11:23
demo       tty2a        May 14 10:13
demo       t0w2         May 17 11:26
demo       t0w1         May 17 11:26
demo       t0w10        May 17 11:26
$
```

Elements of the Menu

The top level of the menu is the ***menu bar***. The menu bar contains items of a general nature. When you select most items on the menu bar, a ***pull-down menu*** is presented which contains items of a more specific nature. Most items on a pull-down menu lead to an action. However, in some cases, an item on a pull-down menu leads to a further menu level called a ***cascading menu***. When some action items are selected, they present a ***dialog box*** which requests further information or confirmation of the action. Each menu has a ***highlighted selection bar*** indicating which item on the menu is currently selected.

Each action item on the menu shows the window command line equivalent commands. These are referred to as ***accelerators***. Because you can accomplish an action more quickly using the window command line, the menu displays these accelerators so that you may learn the command line method of performing common Facet*Term* functions. Remember, "^W" means Ctrl-W (hold down the "Ctrl" key while pressing the "W" key).

Items on pull-down or cascading menus which lead to another menu or a dialog box indicate this by ending with "...".

Using the Menu

To move the highlighted selection bar left or right among items on the menu bar use the left and right arrow keys:

← and →

If no pull-down menu is selected, moving across the menu bar will simply move the highlighted selection bar. If a pull-down menu is selected, moving across the menu bar will cause the next pull-down menu to be selected. If a cascading menu is selected, the left and right arrow keys will simply cancel the cascading menu.

To move the highlighted selection bar up and down among items on a pull-down or cascading menu, use the up and down arrow keys:

↑ and ↓

On any menu, when the selection bar is highlighting the item you would like to select, simply press

RETURN

to select it.

Rather than moving the selection bar, you can directly select menu items by pressing the key whose letter is underlined (or otherwise highlighted) in the item name.

To cancel a menu or dialog box press the space bar or escape key:

␣ or ESC

When you cancel the menu bar, the menu is completely removed and you are once again interacting with the program on the current window.

The menu may be terminated (in which case it will no longer pop up) by pressing the interrupt key when only the menu bar is present (no pull-down menus). If you inadvertently terminate the menu, it may be restarted by using the window command line to select window 10 (or the window you were running the menu in):

CTRL W   0

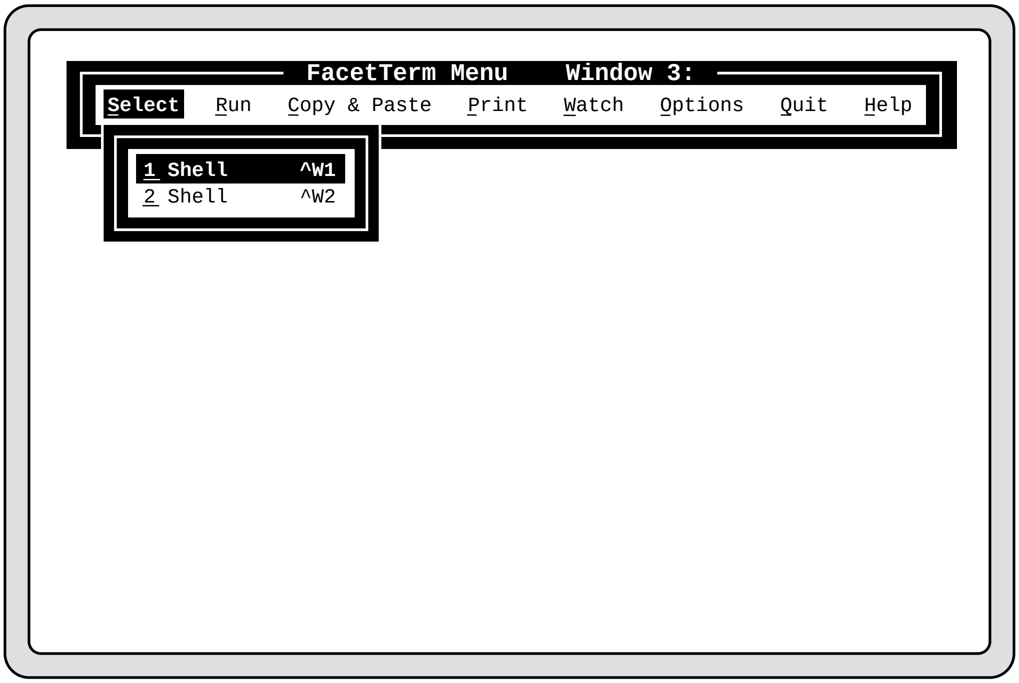
and then using the command line to re-activate that window:

CTRL W   A

Selecting Windows

Selecting Windows from the Facet*Term* Menu

To select an active window, press the menu hot-key to bring up the menu. The menu, as shipped, will automatically pull down the window selection menu:



The window selection menu will show the window number and title of each active window. Move the highlighted selection bar to the window you want to select or type the number of the window you want to select. The menu will be removed, and the window you selected will be refreshed.

Selecting Windows with the Window Command Line

To select a window using the window command line, press the window command line hot-key and then press the number key corresponding to the desired window number (using 0 for window 10). For example, to select window 3, press:

CTRL W   3

To select window 10 press:

CTRL W   0

Scanning Windows

You can also step through the currently active windows (windows that have programs running in them). To do this, press the command line hot-key and then press the “+” key to see higher numbered windows:

CTRL W   +

or press the “-” key to see lower numbered windows:

CTRL W   -

The “+” and “-” keys do not cause Facet*Term* to exit the window command line.   Use the spacebar to exit the window command line when the desired window is located.

If you have a program running on a window that you do not wish to be included in this scan, use the command:

CTRL W   X   I   Y

to exclude the current window, or

CTRL W   X   I   *win#*   Y

where *win#* is the number of the window you want to exclude from the scan.

Returning to the Previous Window

To return to the last window you were on press:

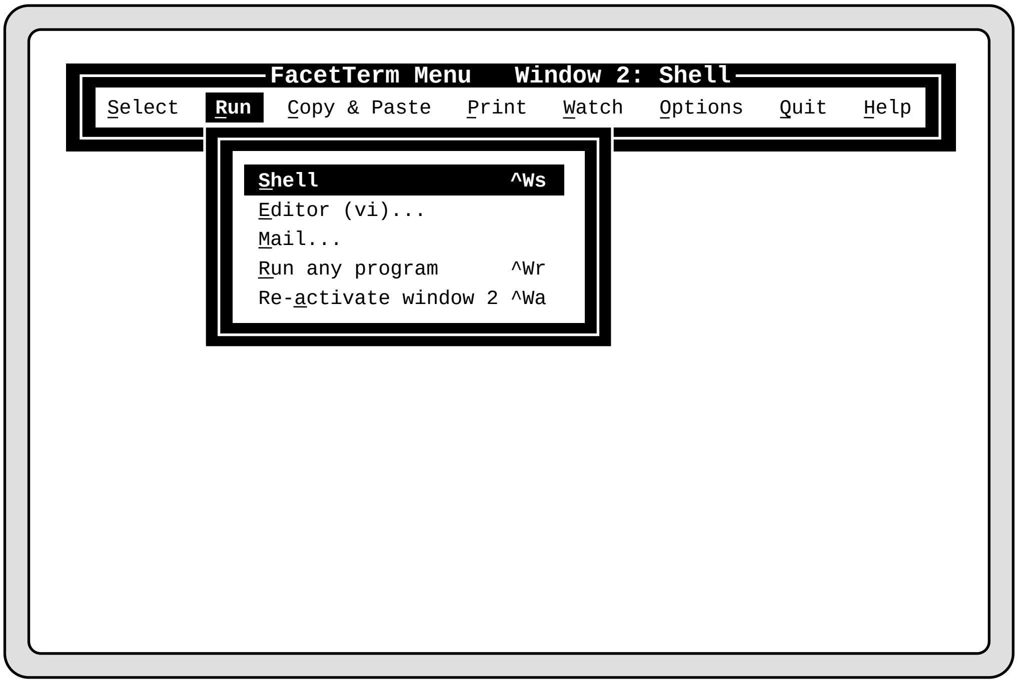
CTRL W   L

For example, if you are working on Window 2, then switch to Window 5, the last window command will put you back on window 2.

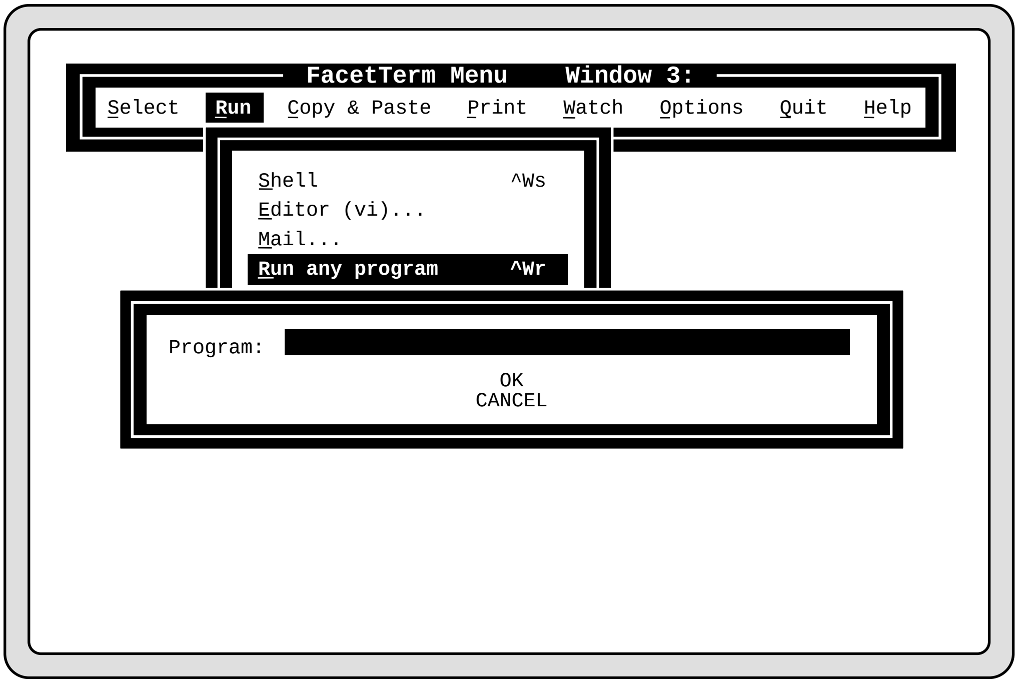
Starting Programs

Starting Programs from the Facet*Term* Menu

To start a program in a new window using the menu, press the menu hot-key to pop up the menu, then move to the "Run" menu and select it. If the program that you want to run is in the list, select it. For example, if you want to start a shell, simply select the "Shell" item on the "Run" pull-down menu:



The program will be started on the next available window. If you want to run a program that is not listed in the menu, select the “Run any program” item. A dialog box will be presented, allowing you to specify the program name and parameters for the program you want started:



After you supply the program command and press Return, then press Return to the "OK" item. The menu will be removed, the next available window will be selected, and your new program will be started there. You can customize the "Run" menu to include your own set of applications to choose from. This is described in the Configuration Guide Chapter in the "Customizing the Facet*Term* Menu" section.

Starting a Shell Using the Window Command Line

To start a new shell on a window, press:

CTRL W   S

If your current window is idle (does not have a program running on it), the program will be run on this window.   If the current window already has a program running on it, Facet*Term* will select the next idle window, switch to it, and run the program there.

The choice of shell is based on the $SHELL environment variable, which indicates the one you normally use.   You may override the value of the $SHELL variable with the $FACETSHELL variable. Your .profile will be executed as if you had just logged in.   You may then run any program you choose, just as you normally do.

Starting Any Program Using the Window Command Line

To start any other programs on a window, press:

CTRL W   R

Facet*Term* will prompt for the name of the program to run.   Type the name of the program and its parameters, if any, just as you would at a shell prompt. You may only specify a single program name. You may *not* specify multiple programs separated by "**;**". You may specify a shell script. Then, press Return, and the next idle window will be selected and the new program will be run there. Pressing Esc, Space or Return without entering a program name cancels the run program command.

Re-activating a Window

If you exit a program in a window and want to re-start it, you can have Facet*Term* re-start it for you by pressing:

CTRL W   A

("A" is for "activate").   If there is an entry in the .facet file for initially starting a program in the window, then the program specified there will be re-started. Otherwise, the first program that was started in that window will be re-started.

Copy and Paste

Facet*Term* allows you to mark an area of characters on a window to be copied into a ***copy buffer***. The characters copied from a window are not removed from the original window. The copy buffer remains intact until another copy operation is performed.

The Paste operation allows you to feed the contents of the copy buffer to an application running in another window, just as though you had typed the characters on the keyboard. Alternatively, you may paste the contents of the copy buffer to a printer or to a file.

Types of Copy Operations

There are several modes of copying data from a window. In all cases, you will mark a beginning and ending location on the window.

A ***block copy*** will copy a rectangle of characters defined by opposing corners that you mark on the window. Trailing spaces will be trimmed from the end of each line. Return characters are inserted between lines. This type of copy is useful for copying whole lines or columns of data out of a table.

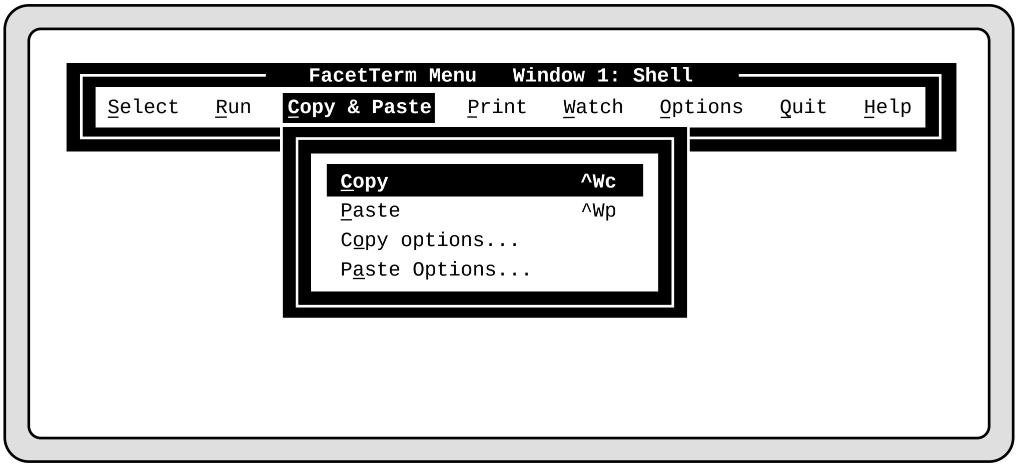
A ***vertical copy*** is like a block copy except that no returns are inserted between lines. Trailing spaces on each line will be collapsed to a single space. This type of copy is useful for copying a long vertical list and submitting it to another program as a string.

A ***stream copy*** will copy all the characters from the first location marked to the last location marked including complete lines in between. Trailing spaces will be trimmed from the end of each line. Return characters will be inserted between each line. This type of copy is useful for copying sentences out of a paragraph.

A ***wrap copy*** is like a stream copy except that no returns are inserted between lines. Leading and trailing spaces on each line will be collapsed to a single space. This type of copy is useful for copying a long shell command that wrapped to multiple lines.

Copy and Paste Using the Facet*Term* Menu

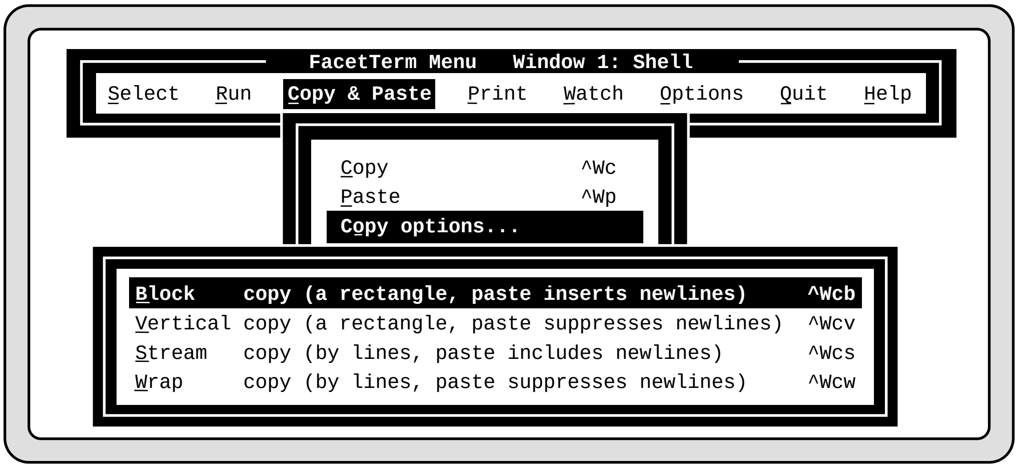
To perform copy and paste functions from the menu, pop up the menu and select the “Copy & Paste” item from the menu bar. A pull-down menu will be presented which contains the various copy and paste functions:



If you choose “Copy” from the pull-down menu, the menu will be removed, and you will be put into the copy operation as if you had entered it from the Window Command Line. Refer to the section below on using the command line to copy.

If you choose “Paste” from the pull-down menu, the menu will be removed and the contents of the copy buffer will be “typed into” the current window that the menu was popped up over.

To change the type of copy, choose “Copy Options”. A cascading menu will be presented which allows you to select the type of copy that you want:


FacetTerm Menu   Window 1: Shell
Select
Run
Copy & Paste
Print
Watch
Options
Quit
Help
Copy ^Wc
Paste ^Wp
Copy options...
Block copy (a rectangle, paste inserts newlines) ^Wcb
Vertical copy (a rectangle, paste suppresses newlines) ^Wcv
Stream copy (by lines, paste includes newlines) ^Wcs
Wrap copy (by lines, paste suppresses newlines) ^Wcw

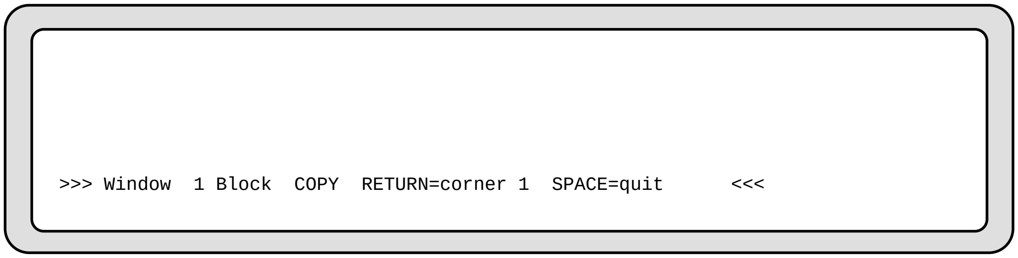
Once you select the type of copy, the menu will be removed. The next time you initiate a copy, it will use the type of copy that has been previously selected.

Using the Window Command Line to Copy

To select the copy operation from the window command line press:

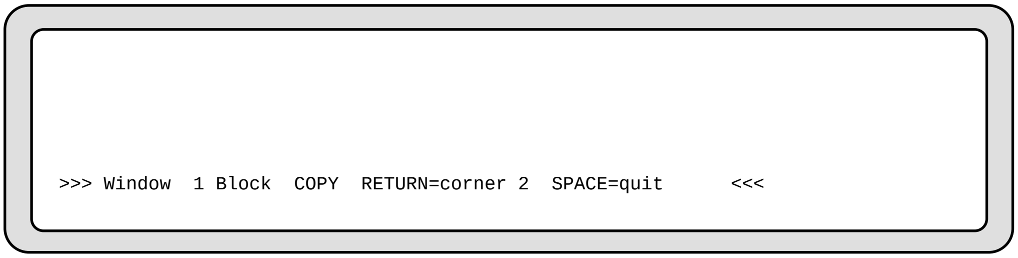
CTRL W   C

Facet*Term* will prompt:



```
>>> Window  1 Block  COPY  RETURN=corner 1  SPACE=quit      <<<
```

Use the arrow keys to position the cursor to the beginning of the text you wish to copy, then press Return. Facet*Term* will then prompt:



```
>>> Window  1 Block  COPY  RETURN=corner 2  SPACE=quit      <<<
```

Use the arrow keys to position the cursor to the end of the text you wish to copy, then press Return. The command line will be removed, and the contents of the window buffer that were marked will be copied into the copy buffer.

Moving the Cursor While in the Copy Operation

While you are marking the text to be copied, there are a variety of keys which you may use to move the cursor:

↑ or U	Move up one character
SHIFT U	Move up to the top row
↓ or D	Move down one character
SHIFT D	Move down to the bottom row
← or L	Move left one character
SHIFT L	Move left to the first column
→ or R	Move right one character
SHIFT R	Move right to the last column
HOME	Move to the first column and first row

Note that you may position the cursor to copy data from the bottom line of your window even though the window command line is obscuring this area while you are in the copy operation.

Marking Corners While in the Copy Operation

While in the copy operation, the window command line will always indicate which corner you are currently marking. The following table shows the keys which are used for marking corners.

RETURN	Mark a corner at the current position
M	Re-mark the first corner at the current position
C	Change which corner is being marked (switch from corner 1 to corner 2 or from corner 2 to corner 1)

Special Commands to Mark Entire Sections

You may either mark the whole screen or a whole line beginning at the current cursor position, without explicitly marking corners. The following keys perform these functions:

A	Marks all characters on the window from the upper left corner to the lower right corner.
E	Marks all characters on a line from the current cursor position to the end of the line.

In both cases, you will still be left in the copy operation. Pressing

RETURN

will finish the copy.

Viewing the Outline of the Copy Area

While you are moving the cursor to mark corners, if you stop pressing keys, the cursor will start moving around the corners of the area to be copied. As soon as you press another key for moving the cursor, the outline movement will stop.

If you would like to have the cursor move one character at a time around the currently marked area press:

O	Causes Facet*Term* to outline the current copy area one character at a time for one cycle around the copy area.

Changing the Type of Copy

At any point during the copy operation, you may change the type of copy. The type of copy being performed is displayed on the command line during the copy operation. The following table summarizes the keys used to change the copy type:

B	Change to a block copy.
V	Change to a vertical copy.
S	Change to a stream copy.
W	Change to a wrap copy.

Using the Window Command Line to Paste to a Window

Once you have used the copy operation to place characters in the copy buffer, they will remain there until you do another copy. To paste the data into a window, select the desired window and make sure that the application running there is ready to receive the data (for example, an editor should be in insert mode rather than command mode). To select the paste operation from the command line, press:

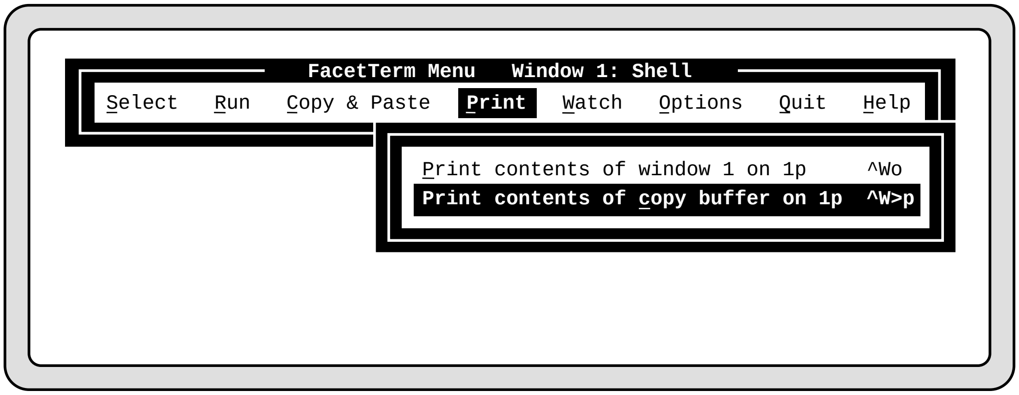
CTRL W P

The characters in the copy buffer will be typed into the current window.

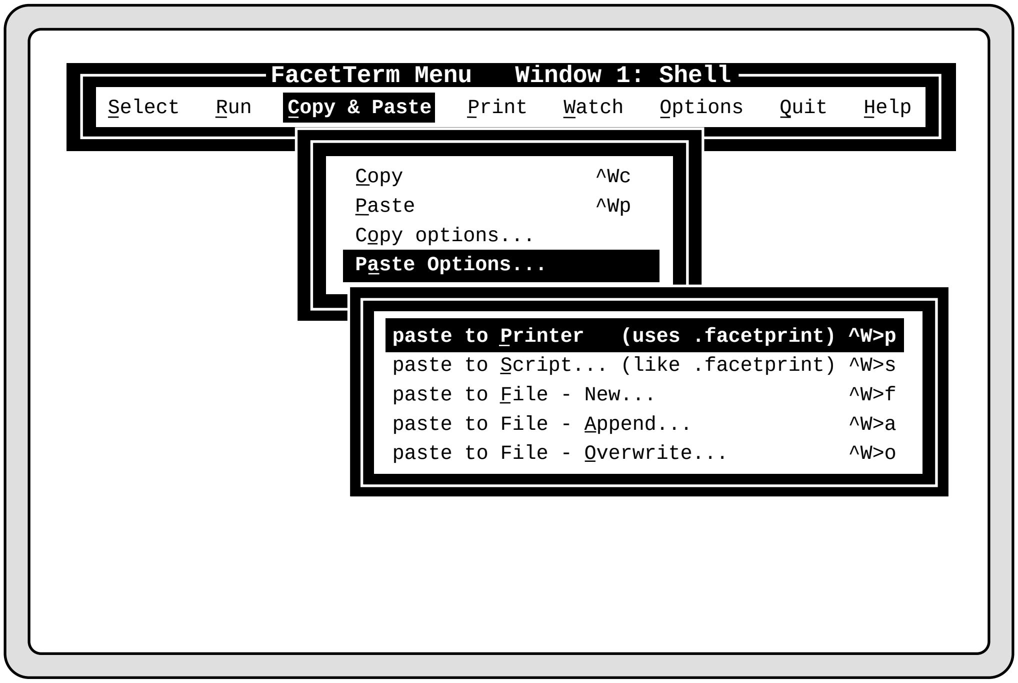
A delay of one second is inserted every 80 characters to prevent the input queues of the destination window from being overrun.

Using the Menu to Print the Contents of the Copy Buffer

To print the contents of the copy buffer using the Facet*Term* menu, pop up the menu and select the "Print" item on the menu bar. A pull-down menu will be presented. Select the item titled "Print contents of copy buffer on lp":


FacetTerm Menu   Window 1: Shell
Select   Run   Copy & Paste   Print   Watch   Options   Quit   Help
Print contents of window 1 on 1p   ^Wo
Print contents of copy buffer on 1p   ^W>p

You may also print the contents of the copy buffer from the "Paste options" cascading menu off of the "Copy & Paste menu":


FacetTerm Menu   Window 1: Shell
Select   Run   Copy & Paste   Print   Watch   Options   Quit   Help
Copy ^Wc
Paste ^Wp
Copy options...
Paste Options...
paste to Printer   (uses .facetprint) ^W>p
paste to Script... (like .facetprint) ^W>s
paste to File - New... ^W>f
paste to File - Append... ^W>a
paste to File - Overwrite... ^W>o

Using the Window Command Line to Print the Contents of the Copy Buffer

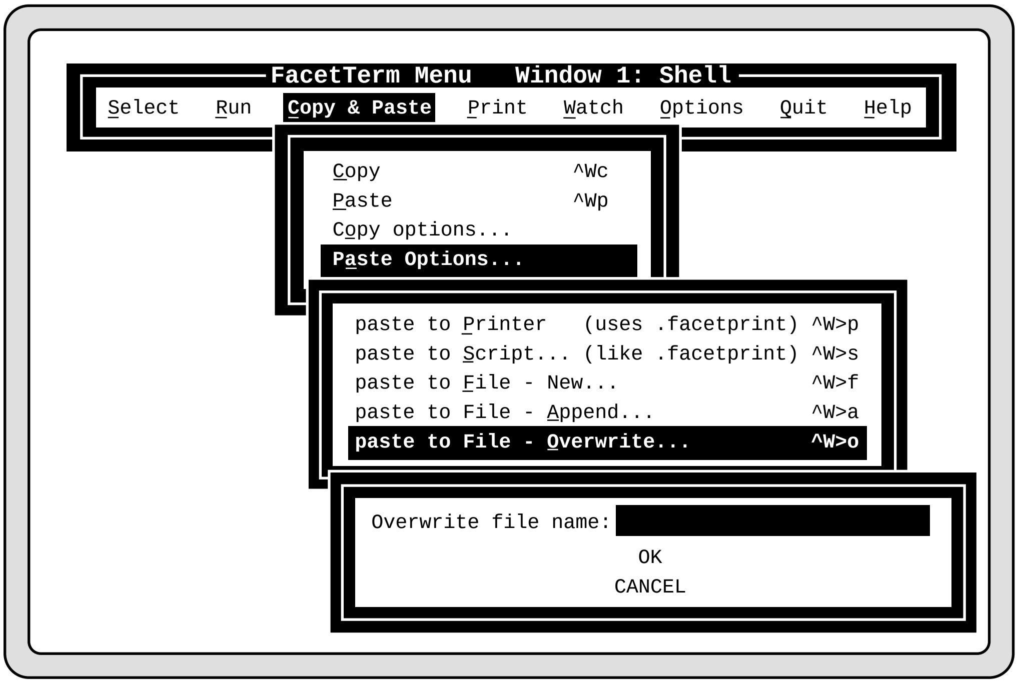
You can use the window command line to print the contents of the copy buffer with the command:

CTRL W   >   P

Whether you use the menu or the command line, the .facetprint file will be used to print the contents of the copy buffer.

Pasting to a File

To use the Facet*Term* menu to paste to a file, pop up the menu, select "Copy & Paste" on the menu bar, select "Paste options" on the pull-down menu, and select the desired paste to file option. A dialog box will prompt you for the name of the file to paste to:


FacetTerm Menu   Window 1: Shell
Select
Run
Copy & Paste
Print
Watch
Options
Quit
Help
Copy ^Wc
Paste ^Wp
Copy options...
Paste Options...
paste to Printer (uses .facetprint) ^W>p
paste to Script... (like .facetprint) ^W>s
paste to File - New... ^W>f
paste to File - Append... ^W>a
paste to File - Overwrite... ^W>o
Overwrite file name:
OK
CANCEL

There are three methods of pasting to a file: new, append, and overwrite. If you choose new, Facet*Term* will not write to the file if it already exists. Append will create the file if it does not exist, and will append to it if it does exist. Overwrite will create the file if it doesn’t exist and will overwrite it if it does exist.

To paste to a file using the window command line, enter:

CTRL W > F

to paste to a new file,

CTRL W > A

to paste append to a file, and

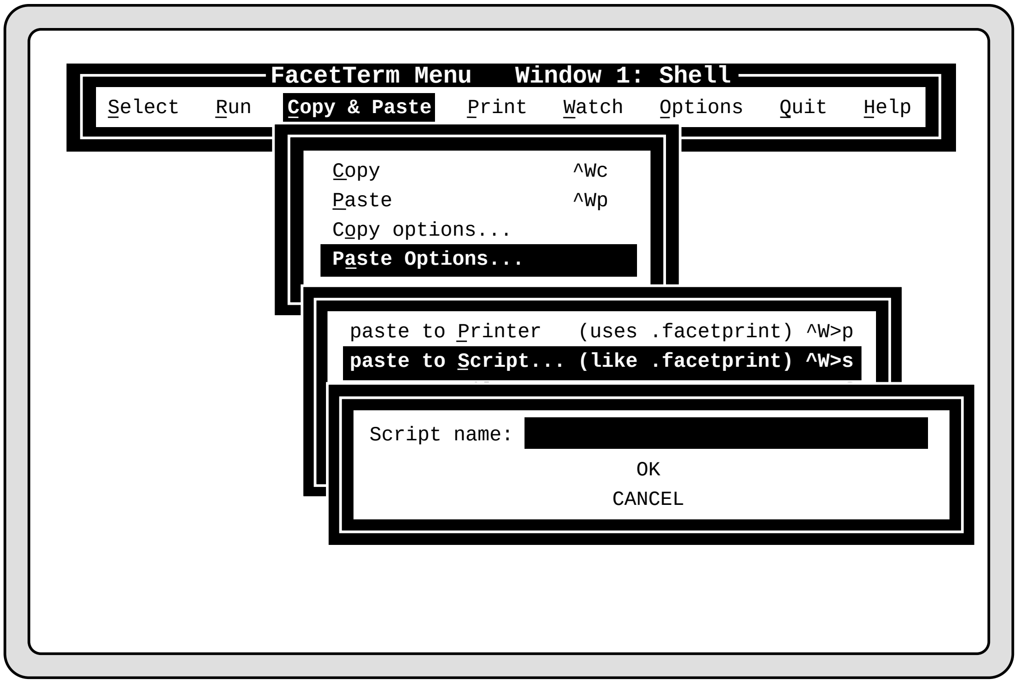
CTRL W > O

to paste overwrite to a file. In all three cases, Facet*Term* will prompt you to enter the name of the file to paste to.

Pasting to a Script

Facet*Term* can send the contents of the copy buffer to a script of your choice, just as it sends it to the .facetprint script in order to print the copy buffer. The contents of the copy buffer will be placed in a temporary file and the name of the file will be given as the first argument to the script. Facet*Term* will beep if it cannot find the script or if it isn't executable. After the script is run, the temporary file will be deleted. The script is searched for along the same path as the .facetprint and .facet files.

This option is found in the Facet*Term* menu for standard terminals under "Copy & Paste", and then "Paste Options". A dialog box will accept the name of the script to paste to:


FacetTerm Menu   Window 1: Shell
Select   Run   Copy & Paste   Print   Watch   Options   Quit   Help
Copy   ^Wc
Paste   ^Wp
Copy options...
Paste Options...
paste to Printer   (uses .facetprint) ^W>p
paste to Script... (like .facetprint) ^W>s
Script name:
OK
CANCEL

Using the window command line, enter:

CTRL W   >   S

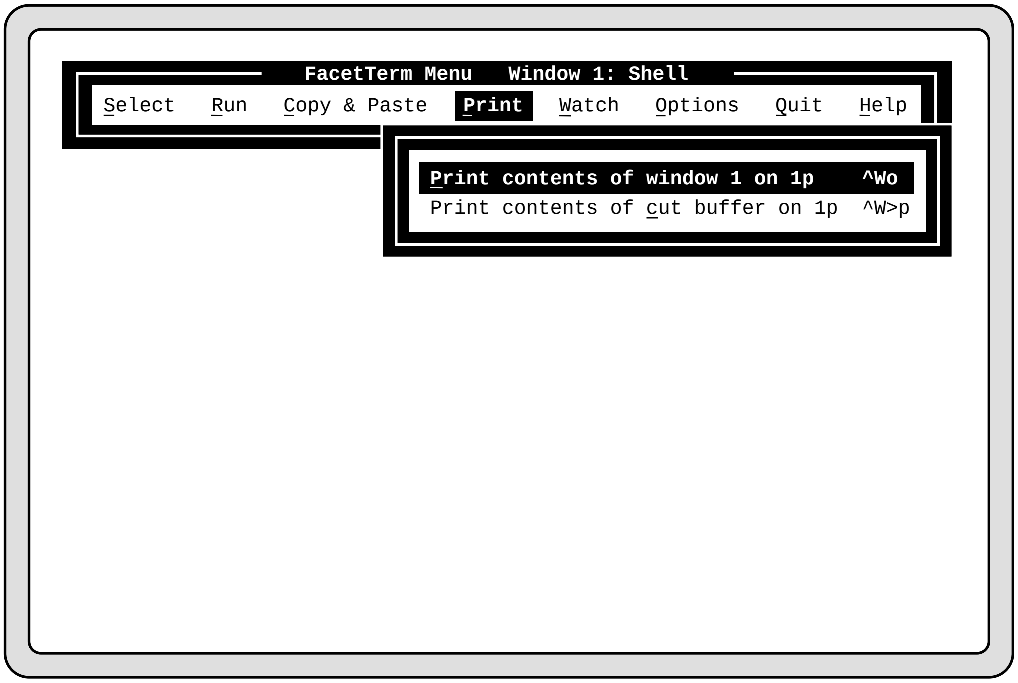
Facet*Term* will prompt for the name of the script.

Printing the Contents of a Window

Facet*Term* allows you to print the contents of a window on the system printer.   The characters in a window are written to a temporary file, and then a shell script, called .facetprint, is executed to print the temporary file. The .facetprint file is discussed in the Configuration Guide chapter.

Selecting Print from the Facet*Term* Menu

To print the current window using the Facet*Term* menu, pop up the menu and select the Print item on the menu bar. A pull-down menu will be presented which displays the print options. Select the item titled "Print screen of window *n* on lp" (where *n* will be the current window number):


FacetTerm Menu   Window 1: Shell
Select
Run
Copy & Paste
Print
Watch
Options
Quit
Help
Print contents of window 1 on 1p   ^Wo
Print contents of cut buffer on 1p ^W>p

Printing a Window Using the Command Line

You may use the window command line to print the current window by pressing:

CTRL W O

(Think of "O" for Output the window to the printer).

Using Facet*Term* Window Watch

Facet*Term* provides a general purpose "Window Watch" feature which can watch for a file to be updated, watch for any activity on a window, watch for specific output to a window, or run a program and test its return code. When a specified event occurs, a visual and/ or auditory "alarm" will notify the user.

One of the most common uses of this feature is to watch for new mail. With the various alarm mechanisms, Facet*Term* can be configured to work with practically any e-mail system and notify the user when he has received new mail.

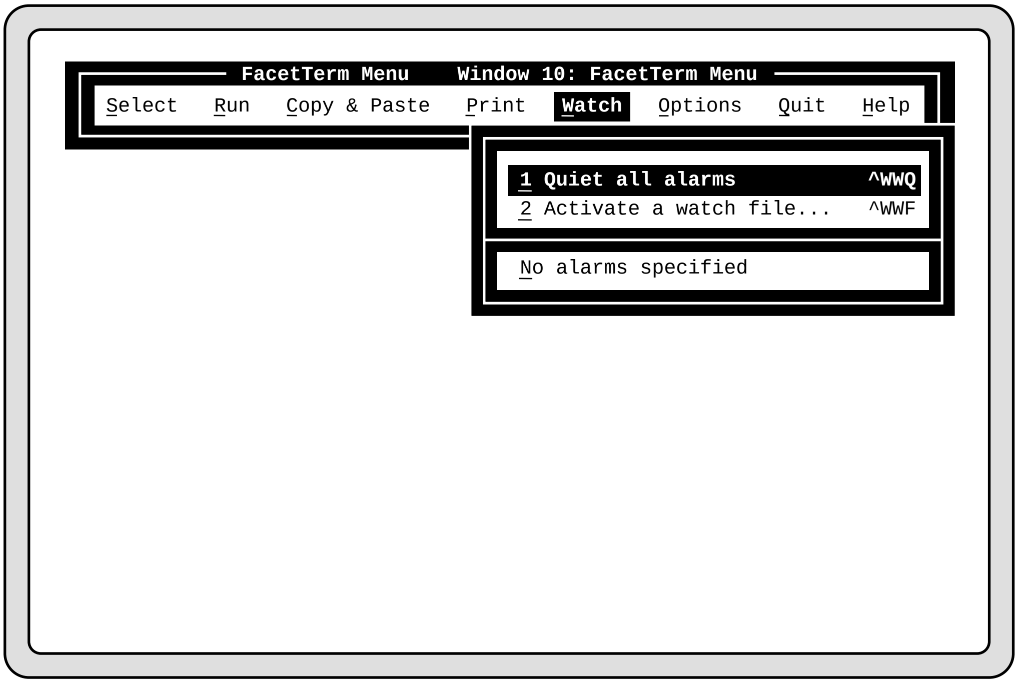
Another common use is to have Facet*Term* watch for a window with a long-running program to output a message when it is through. The user can set the alarm, and then continue working in another window. When the long-running program finishes, the user is notified of its completion.

Watch Files

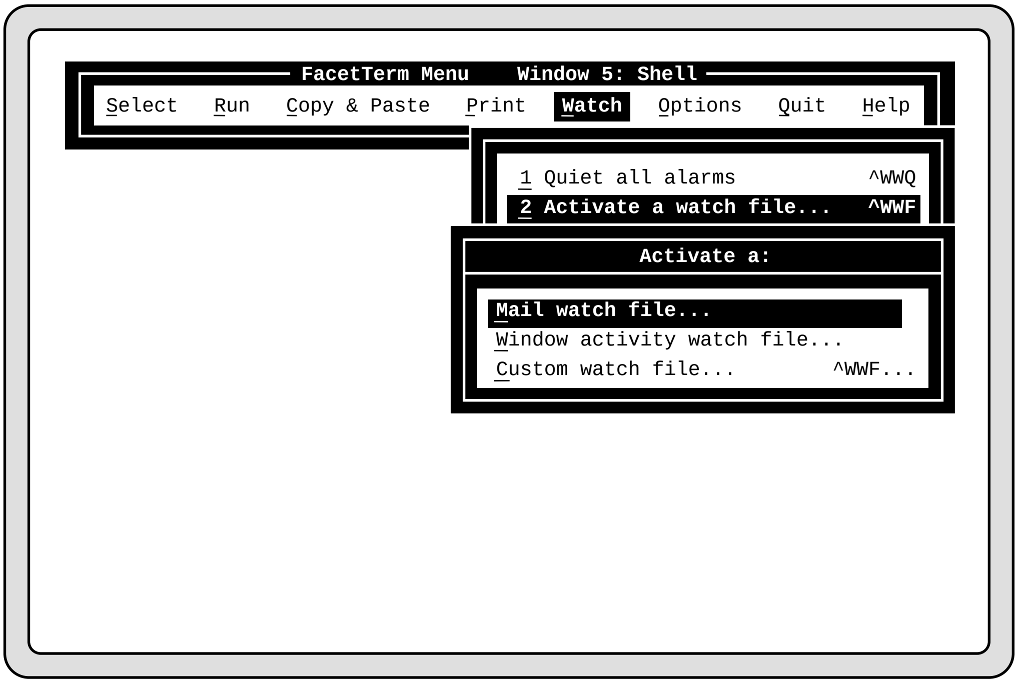
The Window Watch feature is configured with "watch files". The reference for configuring watch files is in the Configuration Guide chapter. Facet*Term* comes preconfigured with a variety of standard watch files that you will find useful for setting up the events you want to watch for.

Using the Facet*Term* Menus to Activate Watch Files

You can create your own watch files or use the ones which have been pre-configured and installed with Facet*Term*. The easiest way to load watch files into Facet*Term* and activate them is via the menus. On a standard terminal, choose the "Watch" item on the menu bar:

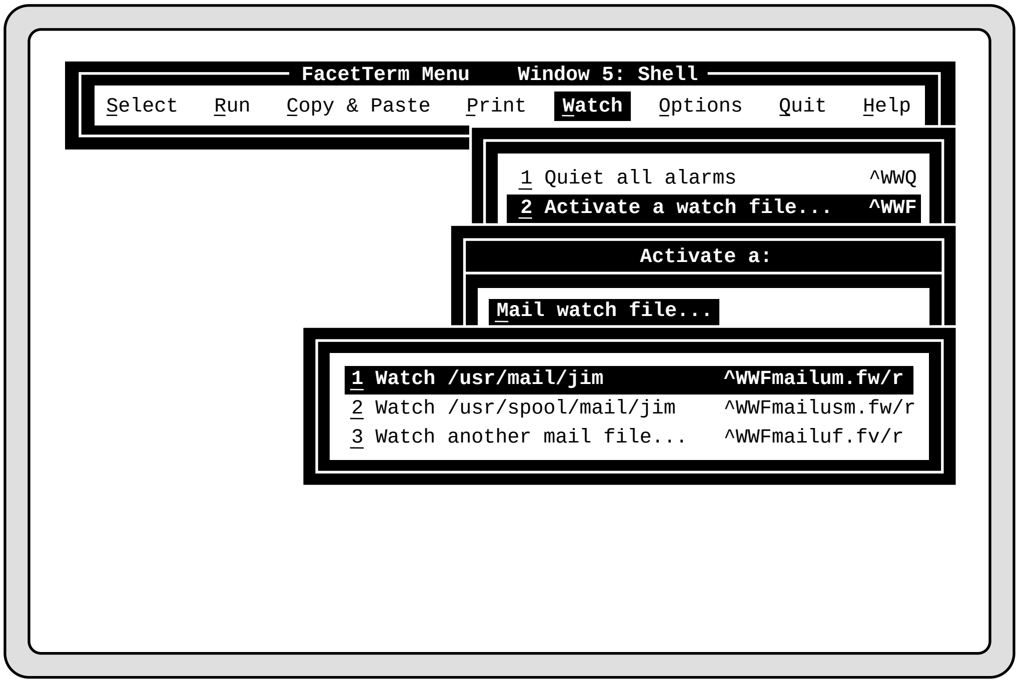

FacetTerm Menu   Window 10: FacetTerm Menu
Select   Run   Copy & Paste   Print   Watch   Options   Quit   Help
1 Quiet all alarms   ^WWQ
2 Activate a watch file...   ^WWF
No alarms specified

Before any watch files have been activated, the "Watch" pull-down menu will show "No alarms specified" in the second section of the menu. If you choose item number 2 on the menu "Activate a watch file", a cascading menu will be presented which will allow you to activate some "canned" watch files supplied with Facet*Term* or activate any watch file of your choosing:


FacetTerm Menu   Window 5: Shell
Select   Run   Copy & Paste   Print   Watch   Options   Quit   Help
1 Quiet all alarms   ^WWQ
2 Activate a watch file...   ^WWF
Activate a:
Mail watch file...
Window activity watch file...
Custom watch file...   ^WWF...

Watching for your mail file to change

If you want to watch for your mail file to change (indicating that you have received new mail), choose the "Mail watch file" item. This will present:

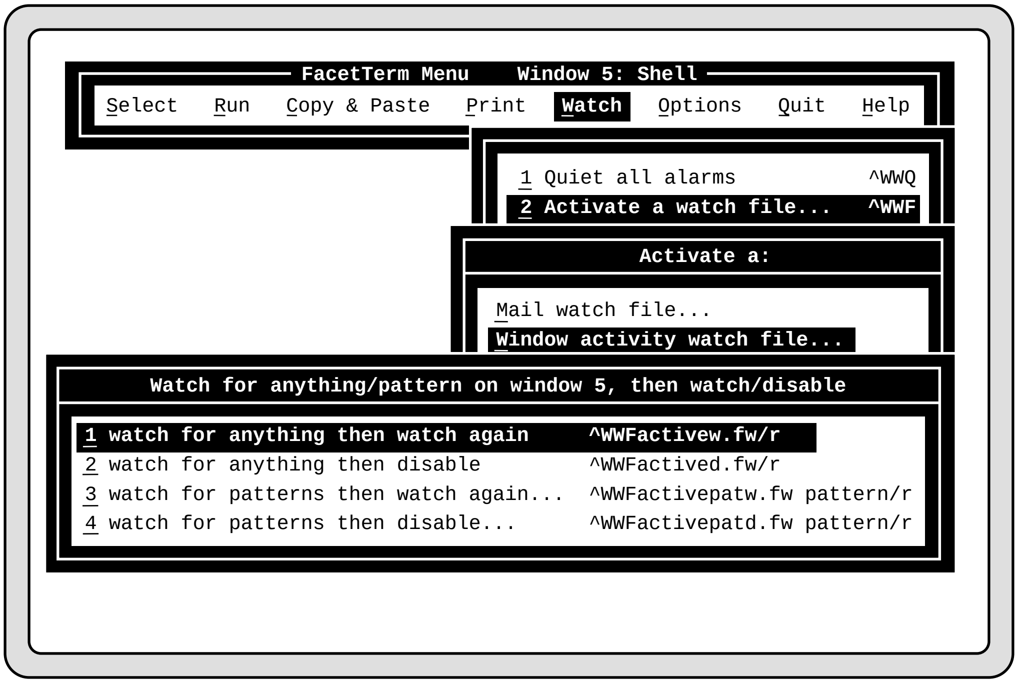

FacetTerm Menu Window 5: Shell
Select Run Copy & Paste Print Watch Options Quit Help
1 Quiet all alarms ^WWQ
2 Activate a watch file... ^WWF
Activate a:
Mail watch file...
1 Watch /usr/mail/jim ^WWFmailum.fw/r
2 Watch /usr/spool/mail/jim ^WWFmailusm.fw/r
3 Watch another mail file... ^WWFmailuf.fv/r

Item 1 on this menu will watch the mail file with your login name in the /usr/mail directory. Item 2 will watch your mail file in the /usr/spool/mail directory. Refer to the documentation on your mail program to determine which is correct for your mail system. If your mail file is named something else, or located in another directory, you may choose item 3 which will give you a dialog box to enter the name of your mail file.

Each of these choices loads the Facet*Term* watch file indicated in the ^W equivalent given with the item (mailum.fw, mailusm.fw, and mailuf.fw). These sample watch files are installed in the /usr/facetterm/text directory.

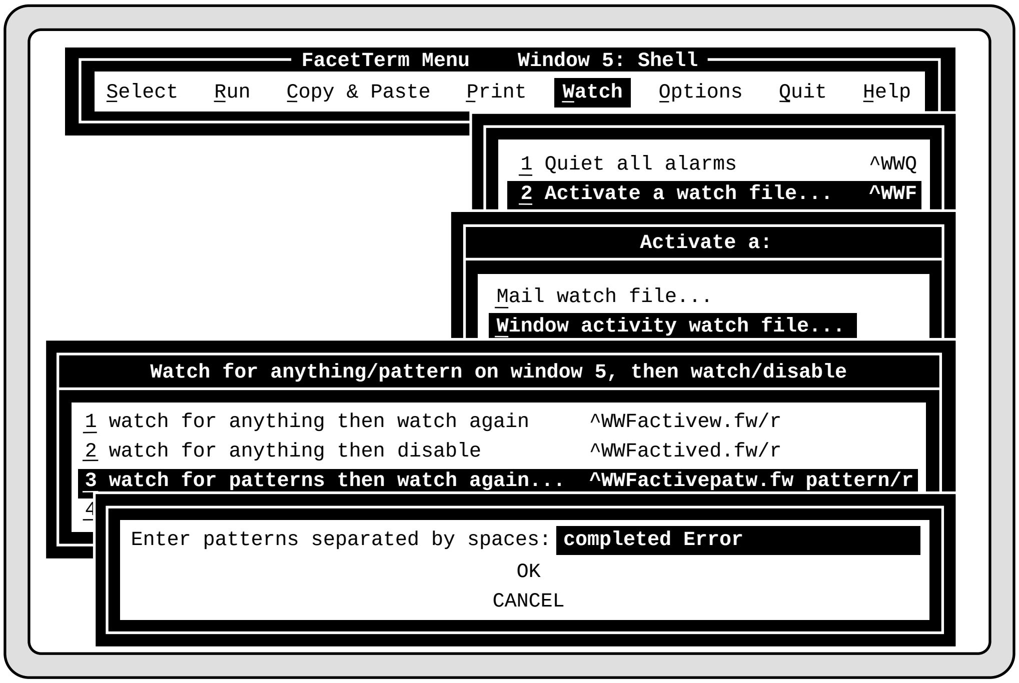
Watching for Any Activity or Specific Messages on a Window

If you want Facet*Term* to watch for any activity (output) or specific messages on a non-current window, choose the item to activate a "Window activity watch file". This will lead to a menu which gives you a choice of 4 watch files to activate:


FacetTerm Menu Window 5: Shell
Select Run Copy & Paste Print Watch Options Quit Help
1 Quiet all alarms ^WWQ
2 Activate a watch file... ^WWF
Activate a:
Mail watch file...
Window activity watch file...
Watch for anything/pattern on window 5, then watch/disable
1 watch for anything then watch again ^WWFactivew.fw/r
2 watch for anything then disable ^WWFactived.fw/r
3 watch for patterns then watch again... ^WWFactivepatw.fw pattern/r
4 watch for patterns then disable... ^WWFactivepatd.fw pattern/r

These four watch files all set up to watch for output on the window that is your current window when you bring the menu up. Items 1 and 2 will activate watch files that look for any output on the window. These two differ in that after output has occurred, and you have switched to that window, item 1 will reset and continue watching for more output, while item 2 will disable the alarm after its first occurrence. Items 3 and 4 load watch files which have Facet*Term* watch for specific messages to be output on the window. These two items also differ in that 3 will reset after an occurrence of the event, and 4 will not.

To have Facet*Term* watch for specific messages, choose either item 3 or 4, and a dialog box will be presented to get the patterns to watch for. You can specify as many patterns as will fit in the space provided. Each pattern must be separated by a space. Patterns cannot have embedded spaces. In this example, we might want to look for "completed" to be output by the program running on the watched window to indicated that it has completed successfully. We might want to watch for "Error" to be output if the program terminates with an error:


FacetTerm Menu   Window 5: Shell
Select   Run   Copy & Paste   Print   Watch   Options   Quit   Help
1 Quiet all alarms   ^WWQ
2 Activate a watch file...   ^WWF
Activate a:
Mail watch file...
Window activity watch file...
Watch for anything/pattern on window 5, then watch/disable
1 watch for anything then watch again   ^WWFactivew.fw/r
2 watch for anything then disable   ^WWFactived.fw/r
3 watch for patterns then watch again...   ^WWFactivepatw.fw pattern/r
4
OK
CANCEL

```
Enter patterns separated by spaces: completed Error
```

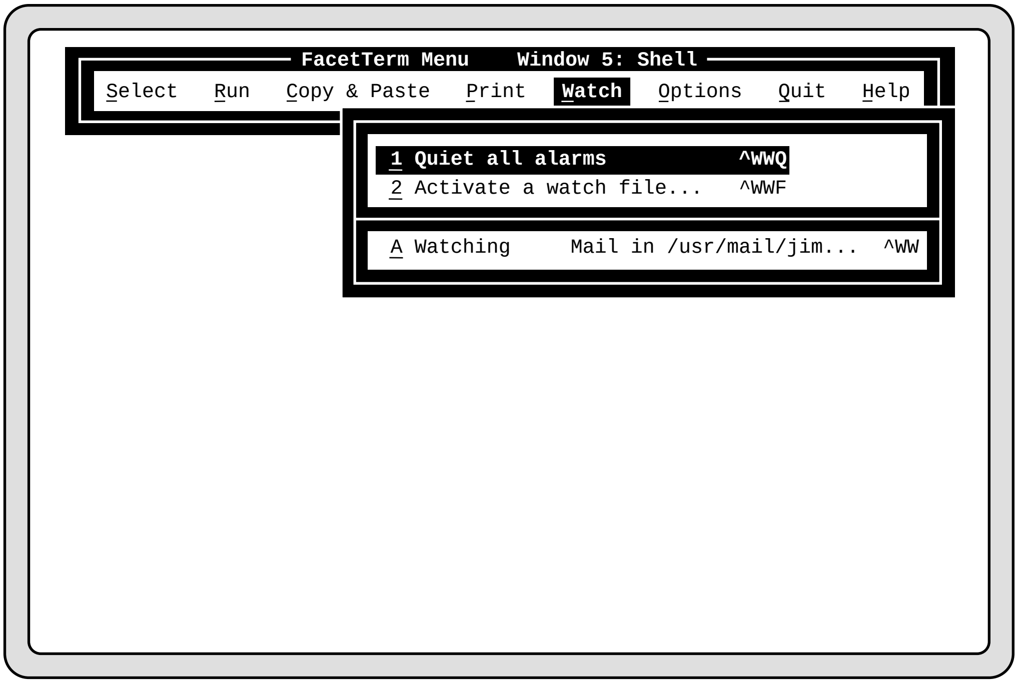
When either the string "completed" or "Error" is output on the watched window, the alarm will go off, letting you know that the program has completed either successfully or with an error.

Acknowledging an Alarm

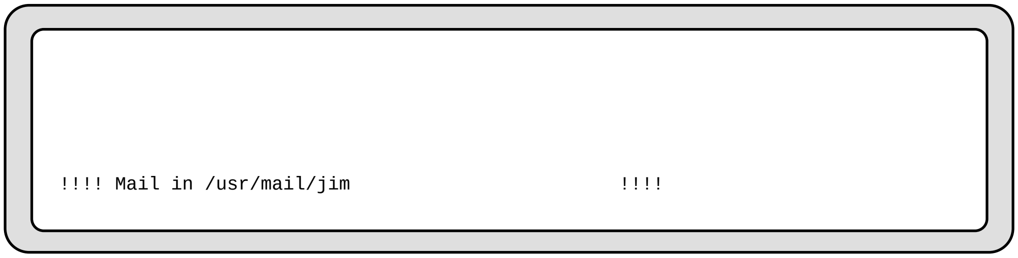
When a watched event occurs, an alarm will go off. Alarms may be configured to be auditory, visual, or both.

The supplied watch files which are accessed through the menu are set-up to give both a visual and auditory alarm.

For example, suppose you have activated a watch on your mail file in the /usr/mail directory. Now when you choose the "Watch" item on the menu, you will see that the bottom half of the pull-down menu shows that this watch is active:

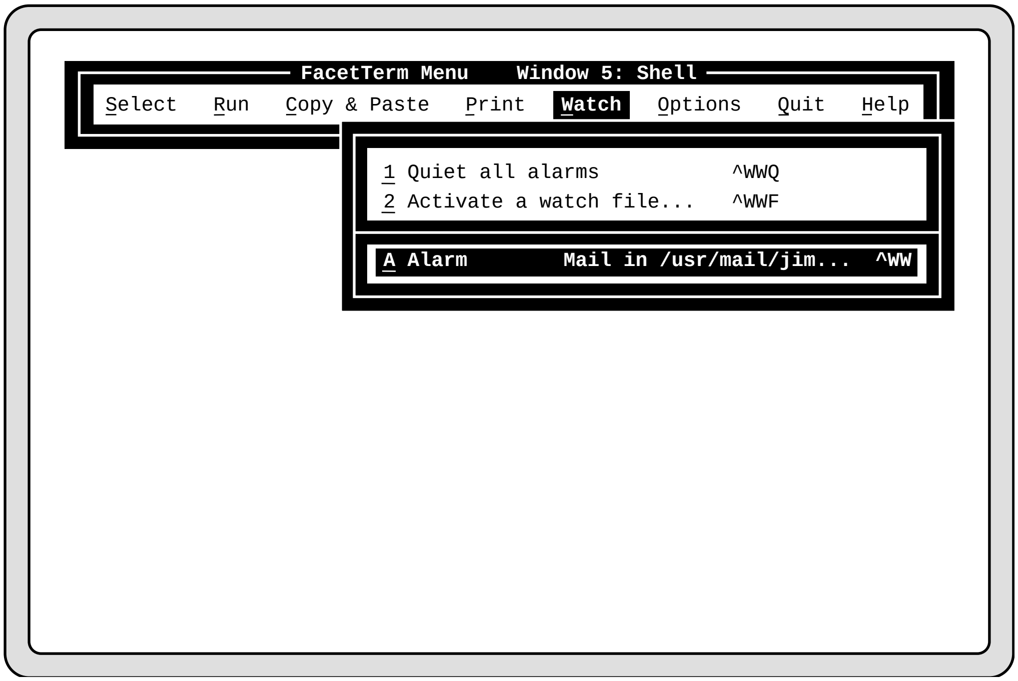

FacetTerm Menu   Window 5: Shell
Select   Run   Copy & Paste   Print   Watch   Options   Quit   Help
1 Quiet all alarms   ^WWQ
2 Activate a watch file...   ^WWF
A Watching   Mail in /usr/mail/jim... ^WW

Watched events are assigned short-cut keys of A-Z. The menu will indicate the state of the watch. In this case it is "Watching" which means that the event has not yet occurred. When new mail arrives in this mail file, Facet*Term* will notify you of this event on the window command line:



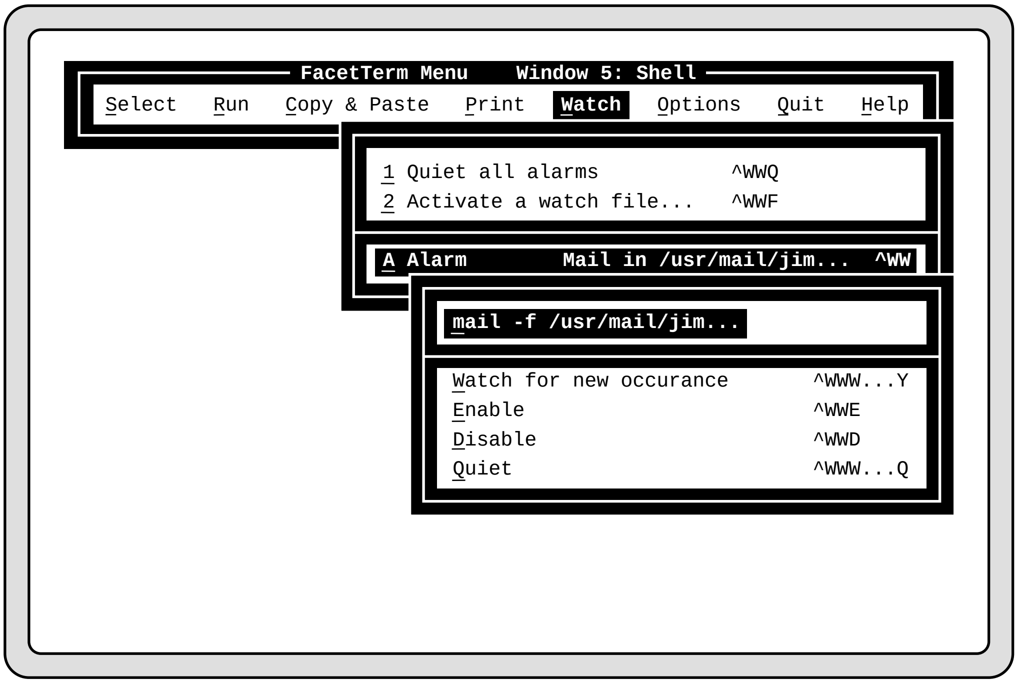
```
!!!! Mail in /usr/mail/jim                        !!!!
```

Now when you choose the "Watch" item on the menu, you will see that the state of the watch has changed to "Alarm":



```
FacetTerm Menu     Window 5: Shell
Select   Run   Copy & Paste   Print   Watch   Options   Quit   Help

                       1 Quiet all alarms           ^WWQ
                       2 Activate a watch file...   ^WWF

                       A Alarm        Mail in /usr/mail/jim...  ^WW
```

When you select the item corresponding to the alarming watch, you will be presented with a further cascading menu which will provide a default action to take:

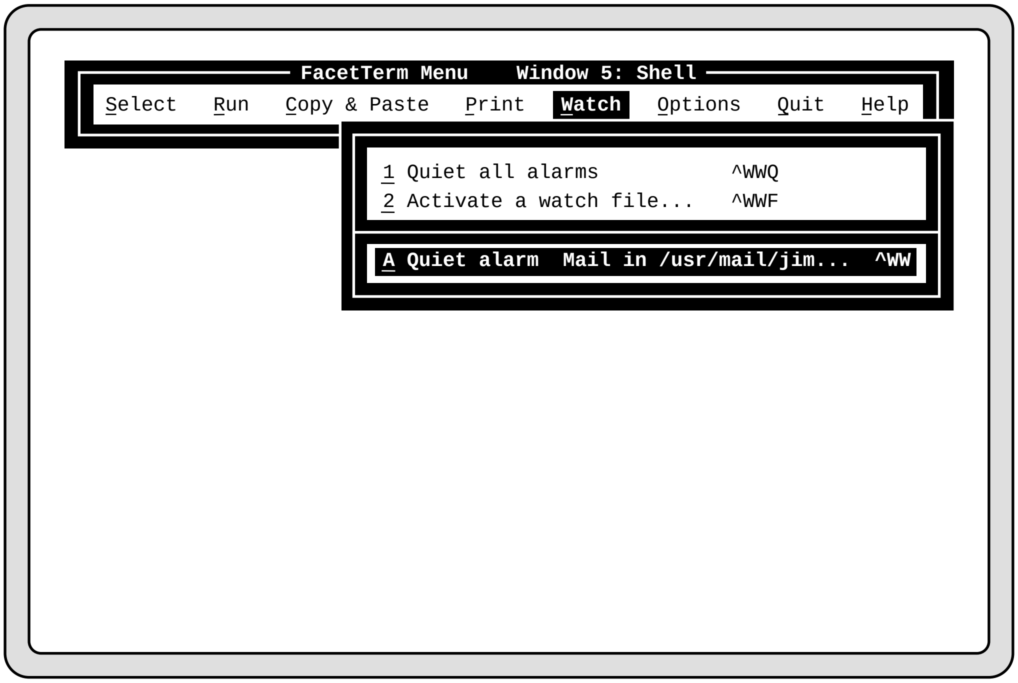


```
FacetTerm Menu    Window 5: Shell
Select   Run   Copy & Paste   Print   Watch   Options   Quit   Help

      1 Quiet all alarms           ^WWQ
      2 Activate a watch file...   ^WWF

      A Alarm        Mail in /usr/mail/jim...  ^WW

            mail -f /usr/mail/jim...

            Watch for new occurance       ^WWW...Y
            Enable                        ^WWE
            Disable                       ^WWD
            Quiet                         ^WWW...Q
```

When watching a mail file, the default action will be to run the mail program to read the mail file. When watching a window for output, the default action will be to select the window being watched as the current window.

The bottom part of this menu provides actions for any watch that has been activated. You can choose "Watch for new occurrence" to acknowledge an alarm without taking the default action. You can "Enable" a previously disabled watch, or "Disable" a currently enabled watch. You may also "Quiet" a watch which is alarming.

You can also quiet all alarms at once by choosing the "Quiet all alarms" item on the "Watch" menu. You may want to do this if you are on the phone or meeting with someone, and find the auditory alarm to be distracting.   Quieting a watch does not reset the watch or disable it, but rather leaves it in a "quiet alarm" state:



```
FacetTerm Menu    Window 5: Shell
Select   Run   Copy & Paste   Print   Watch   Options   Quit   Help

                  1 Quiet all alarms           ^WWQ
                  2 Activate a watch file...   ^WWF

                  A Quiet alarm  Mail in /usr/mail/jim...  ^WW
```

Loading the same watch file each time you start FacetTerm

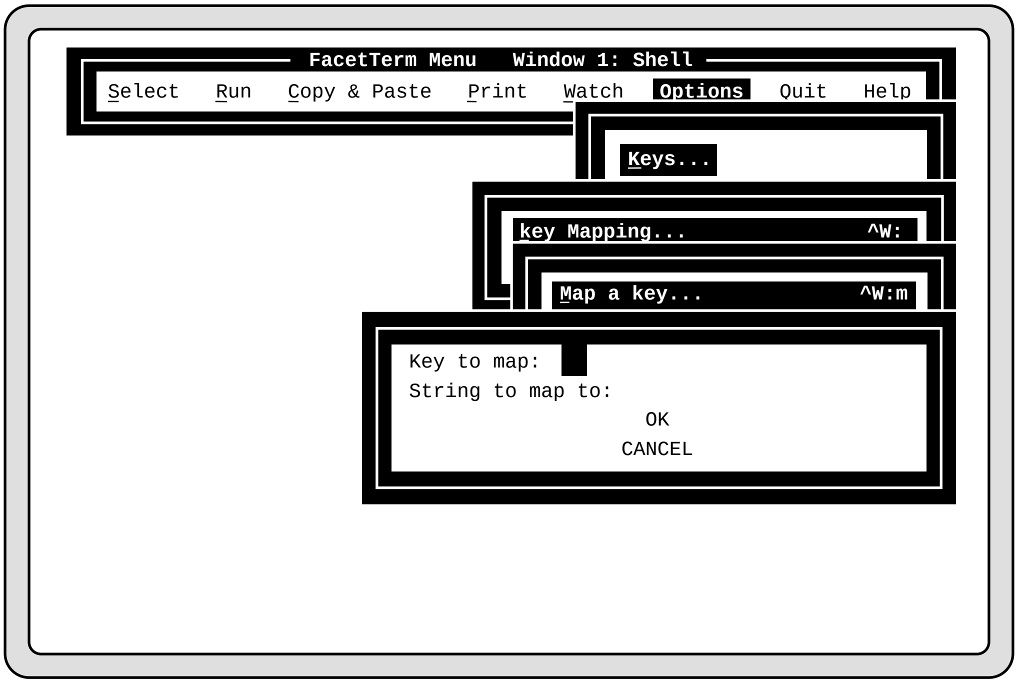
If you want to activate the same watch file(s) each time you start Facet*Term*, you can specify this in your .facet file rather than having to manually load the watch file each time you run Facet*Term*. This procedure is documented in the .facet file reference section in the Configuration Guide chapter. The default .facet file shipped with your software also contains this documentation.

Facet*Term* Key Mapping (Macros)

Facet*Term* provides a simple key mapping or "macro" capability. You can map a single character to a character string to be "typed" in its place. For instance, suppose you are entering new sales leads into a database, and you want to specify the lead source on each entry in the batch to be "Trade Show". Rather than having to type this phrase into the database for each lead, you could program a key (such as Ctrl-L for "Lead") to be a short-cut to type it for you.

Key Mapping Using the Facet*Term* Menu

The key mapping feature is found in the Facet*Term* menu under "Options", then "Keys...", then "Key Mapping...". When you select this item, it will present a dialog box for you to enter the key to be mapped, and the character string to map it to:


FacetTerm Menu   Window 1: Shell
Select   Run   Copy & Paste   Print   Watch   Options   Quit   Help
Keys...
key Mapping...   ^W:
Map a key...   ^W:m
OK
CANCEL

```
Key to map:
String to map to:
```

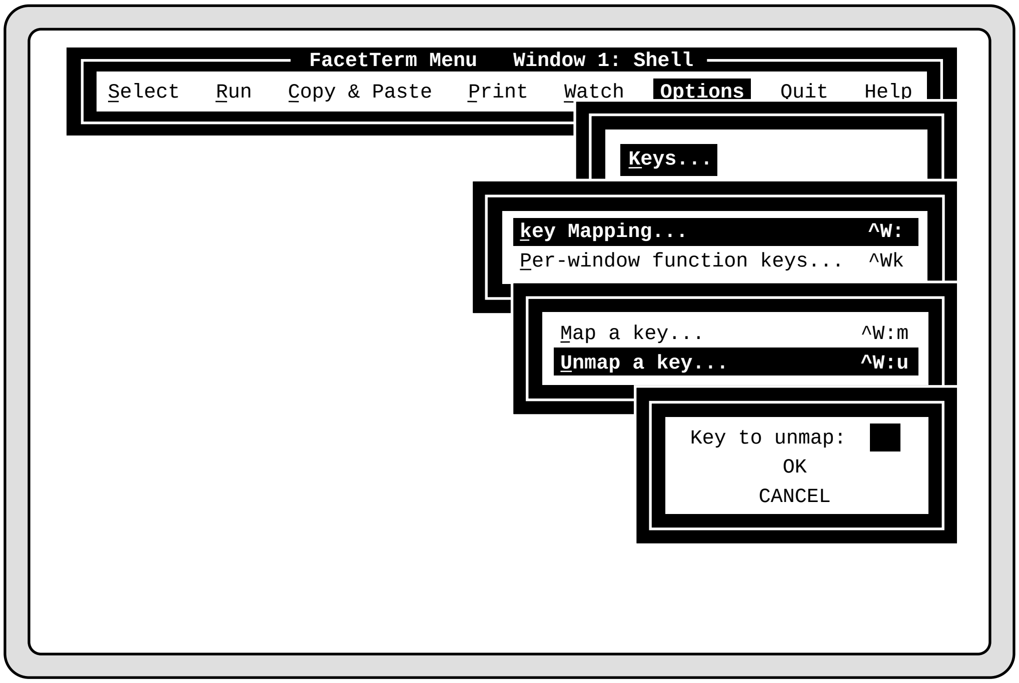
In general, you will only want to map special characters such as control characters. Be careful not to map a control character which has special significance such as Ctrl-M, which is the Return character. When you specify the key to map to either the menu or window command line, indicate a control character with the "^" symbol followed by the character being modified, and don't actually press the control character. For example, if you want to map Ctrl-L, do not enter:

CTRL L

Instead enter:

^ L

To restore a key to its original function, choose the Unmap item on the Key Mapping menu and supply the key to be unmapped to the dialog box:



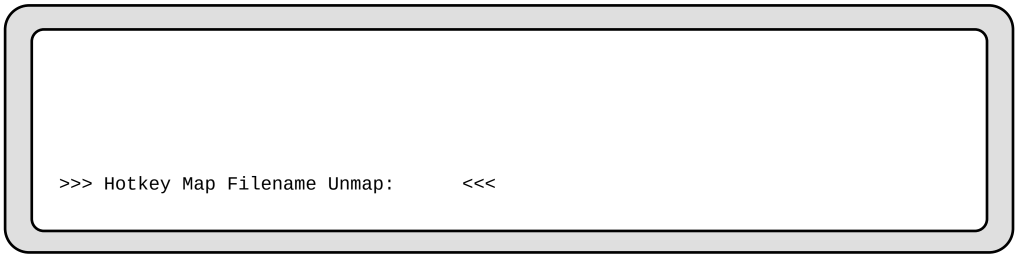
```
FacetTerm Menu   Window 1: Shell
Select   Run   Copy & Paste   Print   Watch   Options   Quit   Help

                                          Keys...

                               key Mapping...               ^W:
                               Per-window function keys...  ^Wk

                                  Map a key...              ^W:m
                                  Unmap a key...            ^W:u

                                        Key to unmap:
                                              OK
                                            CANCEL
```

Key Mapping Using the Window Command Line

To map a key using the window command line, press:

CTRL W :

The command line will prompt:

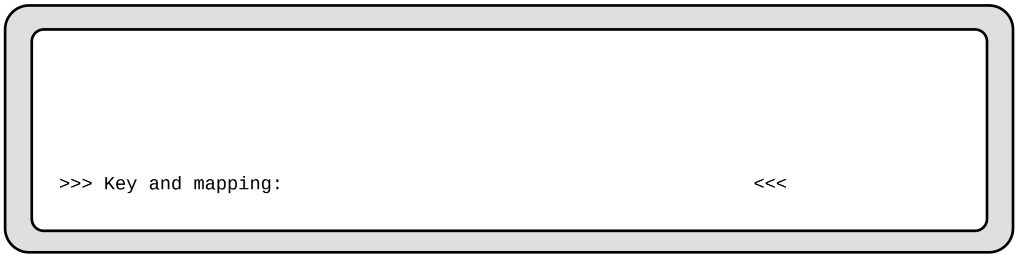


```
>>> Hotkey Map Filename Unmap:      <<<
```

Then select the mapping function by pressing

M

and the command line will prompt:



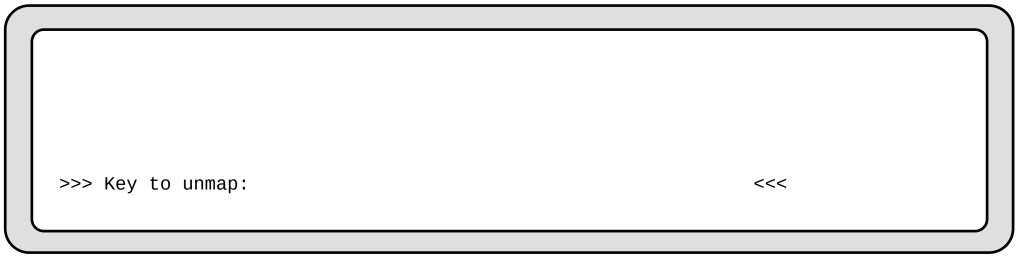
```
>>> Key and mapping:                                          <<<
```

Finally, enter the key to map followed by the string to map it to. Remember, to specify control keys with the "^" syntax. Do not separate the key to be mapped and the string with any spaces. For example, to map Ctrl-G to the string "beep", you would type "^Gbeep" and then Return, to the "Key and mapping" prompt.

To unmap a key with the window command line, press

CTRL W : U

and the command line will prompt:



```
>>> Key to unmap:                                             <<<
```

Then supply the key to be unmapped using the "^" syntax for control keys. For example, to unmap the Ctrl-G key, you would enter "^G" and then Return.

Using Function Keys with Facet*Term*

Note: If your terminal does not have programmable function keys (such as a VT100) then this section does not apply to you.

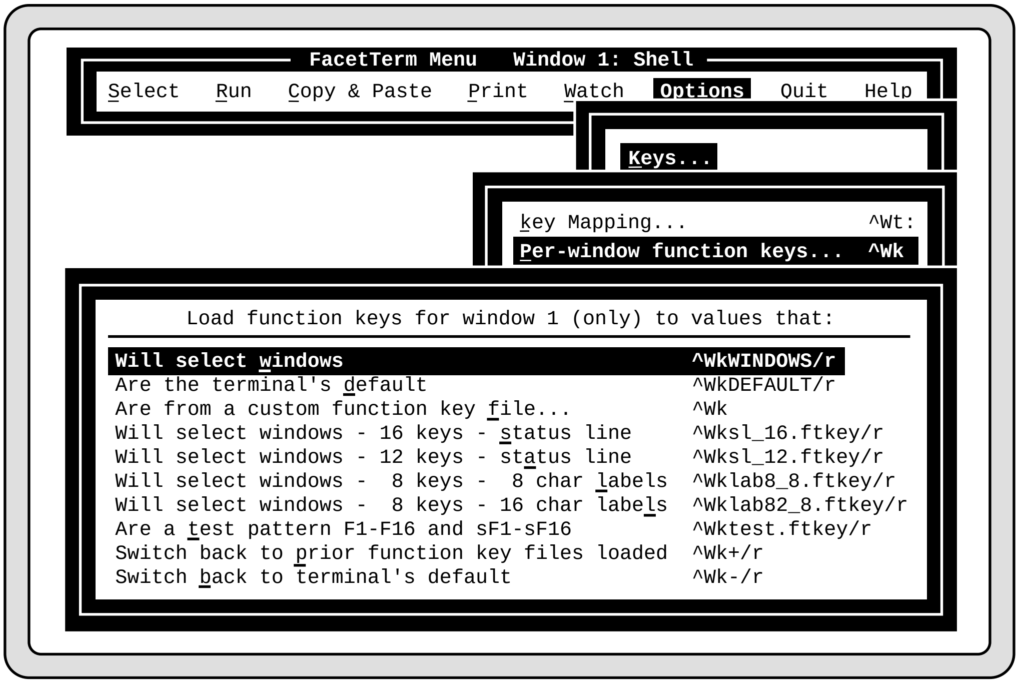
Facet*Term* is quite flexible in its handling of function keys. Because many applications use every function key available on the terminal, Facet*Term* does not, by default, use function keys itself. Instead, it watches for your applications to program function keys, and switches this programming on a per-window basis.

If your applications do not use function keys, Facet*Term* can use them to select windows, pop up the menu, bring up the window command line, etc. This is done by telling Facet*Term* to load a function key file. Several function key files are included with Facet*Term*. You may also create your own special purpose function key files. The format of these files is explained in the Configuration Guide Chapter in the section titled "Making Facet*Term* Function Key Files". Function key files may be either loaded for a particular window or globally for all windows.

For each window, Facet*Term* remembers the last function key file that was loaded, the default values of the function keys for your terminal, and the last values programmed by an application running on a window.

Using the Facet*Term* Menu to Manage Function Keys

You may use the menu to load function key files. Select "Options" from the menu bar, then "Keys...", and you will see a pull-down menu which includes items for setting up function keys on a per-window basis, or globally for all windows. Each of these items leads to a cascading menu which contains numerous "built-in" selections for having Facet*Term* use function keys. If you choose the "Per-window function keys" item you will see:



```
FacetTerm Menu   Window 1: Shell
Select   Run   Copy & Paste   Print   Watch   Options   Quit   Help

                                          Keys...

                                     key Mapping...              ^Wt:
                                     Per-window function keys...  ^Wk

      Load function keys for window 1 (only) to values that:

Will select windows                                ^WkWINDOWS/r
Are the terminal's default                         ^WkDEFAULT/r
Are from a custom function key file...             ^Wk
Will select windows - 16 keys - status line        ^Wksl_16.ftkey/r
Will select windows - 12 keys - status line        ^Wksl_12.ftkey/r
Will select windows -  8 keys -  8 char labels     ^Wklab8_8.ftkey/r
Will select windows -  8 keys - 16 char labels     ^Wklab82_8.ftkey/r
Are a test pattern F1-F16 and sF1-sF16             ^Wktest.ftkey/r
Switch back to prior function key files loaded     ^Wk+/r
Switch back to terminal's default                  ^Wk-/r
```

The first selection on the cascading menu "Will select windows" is a generic choice which loads a function key file that will allow you to select windows using function keys in the best way possible for your terminal. In general, F1 will select window 1, F2 will select window 2, etc. A notable exception is the VT220 and work-alikes that have F6 as the first programmable key. On these terminals, F6 selects window 1, etc. The number of keys which may be used to directly select windows depends on the number of function keys on your terminal. The last 4 function keys will be programmed to select the last window, scan through active windows, bring up the window command line, and pop up the menu. The selections such as "Will select windows - 16 keys - status line", etc. allow you to explicitly tell Facet*Term* how to use your function keys and whether you have a status line or function key labels for displaying the meaning of each of the programmed function keys.

The selection "Are the terminal's default" allows you to set the function keys to the default values for your type of terminal. This setting is remembered by Facet*Term* as the last function key file loaded on the window.

The selection "Are from a custom function key file..." allows you to set the function keys to values specified in a custom function key file which you supply. This selection will present a dialog box to get the name of the function key file.

The selection "Are a test pattern F1-F16 and sF1-sF16" is for testing purposes only. This selection will program function key F1 to transmit "F1", key F2 will transmit "F2", etc.

The last two selections "Switch back to prior function key file loaded" and "Switch back to terminal's default" allow you to switch between the values specified in the last function key file loaded and the terminal's default values. They also allow you to restore the terminal to either the last function key file settings or the default settings if an application running in a window has programmed the keys to something else.

Going back to the "Keys..." cascading menu, the other function key item is "Global function keys". This selection allows you to do all of the above except that it is done globally to all windows at once rather than only to the current window.

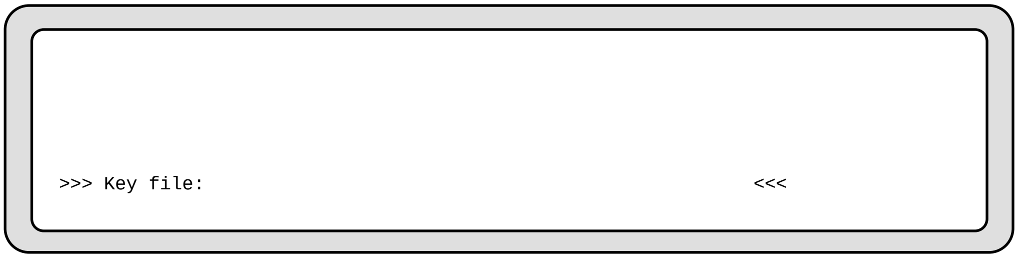
Using the Window Command Line to Manage Function Keys

As indicated on the menu, there are window command line commands to do all of the functions explained above.

To load function key settings for the current window only press:

CTRL W   K

Facet*Term* will prompt:



```
>>> Key file:                                              <<<
```

You may enter the name of your own custom function key file or one of the following special function key files that are included with Facet*Term*:

DEFAULT	To load the terminal's default values.
WINDOWS	To load the function keys to select windows in a manner that is appropriate for your terminal.
sl_16.ftkey	To load the function keys to select windows for a terminal with 16 function keys and a status line.
sl_12.ftkey	To load the function keys to select windows for a terminal with 12 function keys and a status line.
lab8_8.ftkey	To load the function keys to select windows for a terminal with 8 function keys and 8 character labels for each key.
lab82_8.ftkey	To load the function keys to select windows for a terminal with 8 function keys and 16 character labels for each key (usually 2 line labels with 8 characters on each line).
+	To restore the function keys to the values last loaded from a function key file.
-	To restore the function keys to their default values without disturbing Facet*Term*'s memory of the last function key file loaded.

Similarly, you may instruct Facet*Term* to load the function keys for all windows by pressing:

CTRL W G

You may provide the same names indicated in the per-window function key programming above.

Using Special Keys with Facet*Term*

In most cases, when you are working in a Facet*Term* window it is no different than when you are working on your terminal without running Facet*Term*. However, there are a few keys which behave differently. Of course, the menu and window command line hot-keys must be pressed twice in order to send them to an application. However, you will probably choose hot-keys which do not conflict with the normal running of your applications.

A more likely conflict is the use of

CTRL S

and

CTRL Q

These keys are used by Facet*Term* to suspend and resume output to the terminal in general, not just on a particular window. Therefore, if you suspend output, you cannot switch windows or interact with FacetTerm in any way until you resume output. If you have a need to send a ^S or ^Q to an application in a window you may do so by pressing:

CTRL W . S

to send a ^S and

CTRL W . Q

to send a ^Q.

If your computer and terminal can use hardware flow control, the Facet*Corp* support staff can help you disable Facet*Term*'s use of ^S and ^Q.

You may also need to send a break or a null character to a program running in a window. Either of these will cause the Facet*Term* window command line to come up. Pressing a break again will cause the break condition to be sent to the current window. (Note: some operating systems do not provide this capability.) You may also send a break to a window by pressing:

CTRL W . B

and you may send a null to a window by pressing:

CTRL W . @

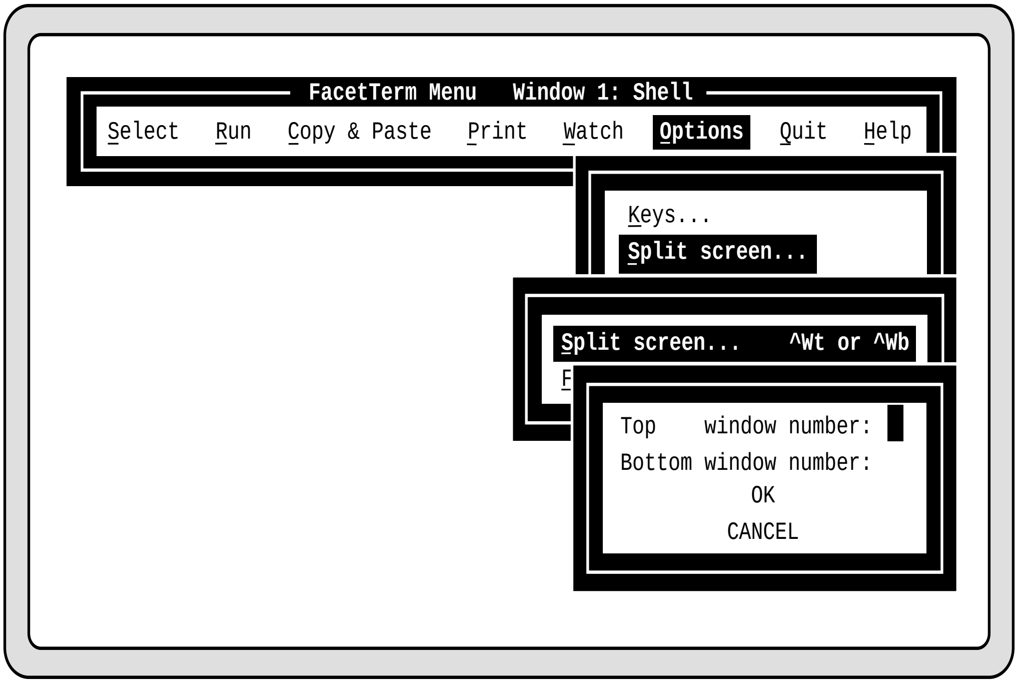
Split Screen

Facet*Term*'s split screen mode divides the terminal's screen horizontally and displays one window in the top half and another window in the bottom half.

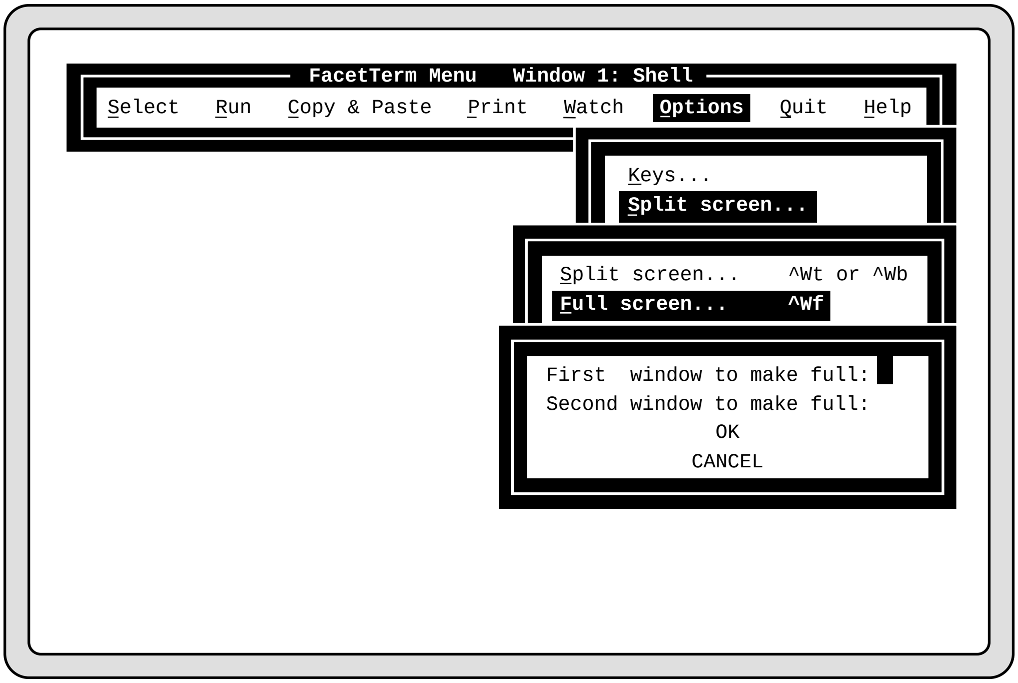
When in split screen mode, Facet*Term* will always pan the window to keep the cursor visible. With some applications this can be quite distracting, making split screen mode unusable for those applications. With other applications (such as a text editor), you may be able to specify to the application that you have fewer lines, matching the size of your split screen window.

Setting Up Split Screens With the Menu

Split screen mode is most easily set up using the menu because you can specify both the top and bottom windows at one time. Bring up the menu and select "Window" on the menu bar. Choose "Split screen" on the pull-down menu, and a dialog box will be presented where you can enter the window numbers of the windows to be put in the top and bottom of the split screen:


FacetTerm Menu   Window 1: Shell
Select   Run   Copy & Paste   Print   Watch   Options   Quit   Help
Keys...
Split screen...
Split screen...   ^Wt or ^Wb
F
Top   window number:
Bottom window number:
OK
CANCEL

You may also use the menu to return the two windows that are in the split back to full screen windows:


FacetTerm Menu   Window 1: Shell
Select   Run   Copy & Paste   Print   Watch   Options   Quit   Help
Keys...
Split screen...
Split screen...   ^Wt or ^Wb
Full screen...   ^Wf
First window to make full:
Second window to make full:
OK
CANCEL

Using the Window Command Line to Manage Split Screens

The following commands are used to manage split screen windows with the window command line. All of these commands leave Facet*Term* in window command mode.

To change a window to be in the top half of a split screen, press:

CTRL W   T

If you then select a window number, that window will be put in the top half of the split screen. If you press Return, the current window will be put in the top half of the split screen.

To change a window to be in the bottom half of a split screen, press:

CTRL W   B

To change a window to be full screen, press:

CTRL W   F

To move the spit screen divider down (giving more lines to the top window and fewer lines to the bottom window), press:

CTRL W   D

or

CTRL W   ↓

Each press of the "D" or down arrow key will move the split down one line. You may press it repeatedly to move the split down multiple lines.

To move the split screen divider up (giving more lines to the bottom window and fewer lines to the top window), press:

CTRL W   U

or

CTRL W   ↑

Each press of the "U" or up arrow key will move the split up one line. The minimum size of either window is two lines.

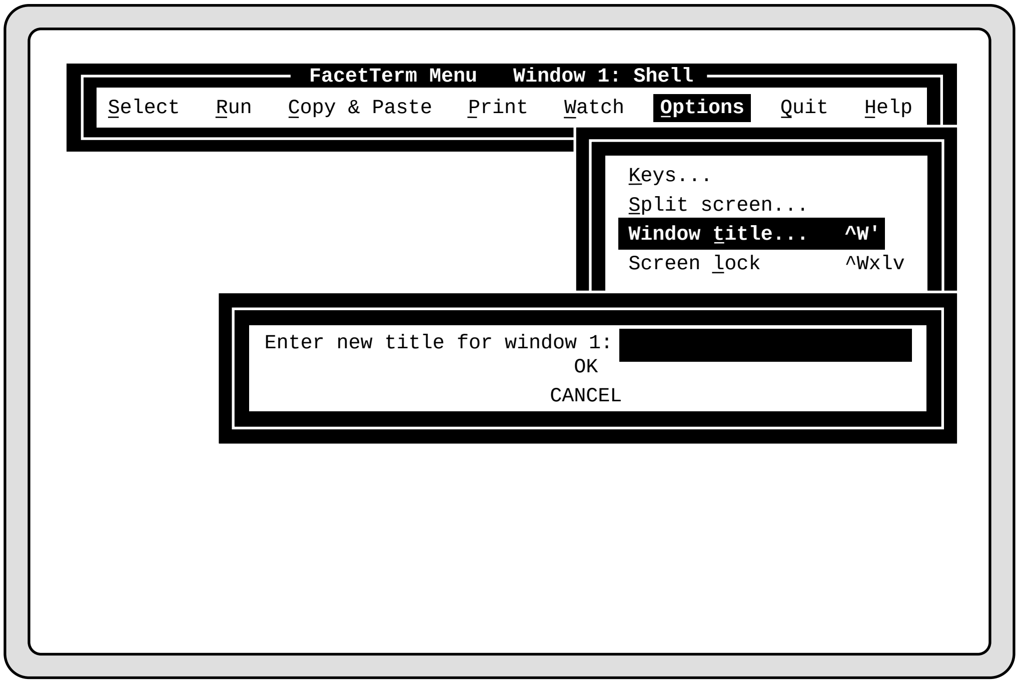
When the split screen divider is where you want it, you may exit the window command line by pressing:

□   or   ESC

Window Titles

Each Facet*Term* window has a title associated with it. These titles are displayed in the window selection menu. Titles are usually assigned in the .facet file for programs being started automatically, or by the menu for programs which were started from the run menu. Occasionally, you may want to change the title of a window.

You can change a window's title from the Facet*Term* menu by selecting the "Options" item on the menu bar, and then "Window Title..." on the pull-down menu. You will be provided with a dialog box in which to enter the new title for the current window:


FacetTerm Menu Window 1: Shell
Select Run Copy & Paste Print Watch Options Quit Help
Keys...
Split screen...
Window title... ^W'
Screen lock ^Wxlv
OK
CANCEL

Enter new title for window 1:

You may change the title of the current window using the window command line by pressing:

CTRL W "

or

CTRL W '

Facet*Term* will then prompt for the new title.

Screen Saver and Screen Lock

Screen Saver

The screen saver feature is for use on PC consoles or terminals which don't have a built-in screen saver. If you are using a terminal which has its own screen saver, you should use it instead.

The screen saver is primarily controlled by the start-up configuration in the .facet file. See the ".facet File Reference" section of the "Configuration Guide" chapter. However, you can activate the screen saver while Facet*Term* is already running from the window command line by pressing:

[CTRL][W] [X] [S] [Y]

You can disable the screen saver by pressing:

[CTRL][W] [X] [S] [N]

When the screen saver is active it clears the screen, and moves the string "Facet*Term* - press any key to continue" around on the screen. You can configure this text string, as well as the time-out period in the .facet file.

Screen Lock

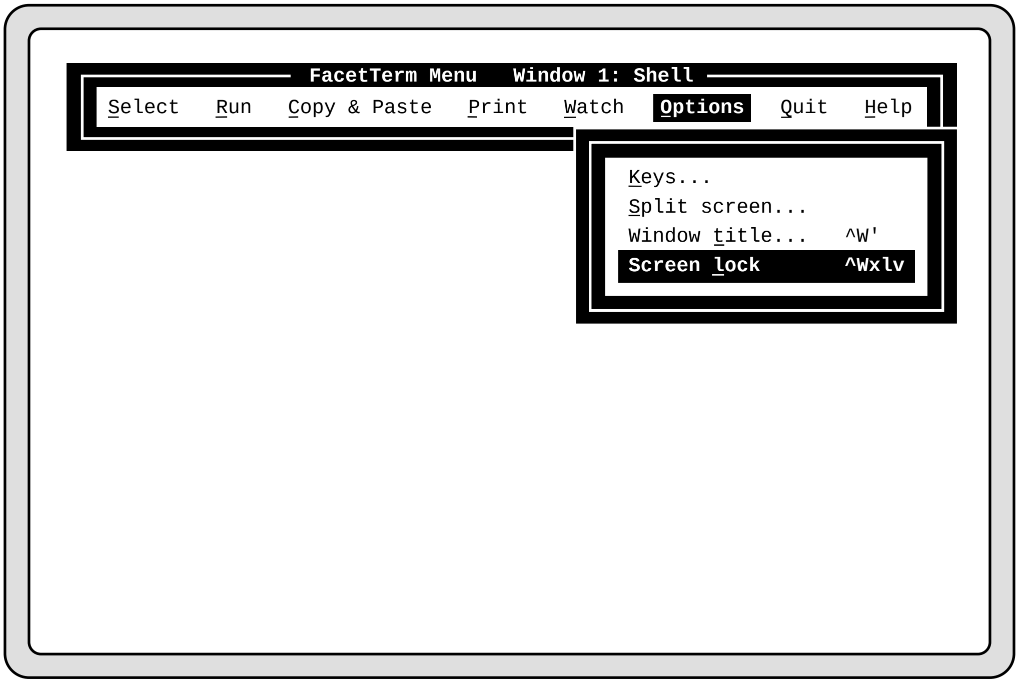
Facet*Term* provides a security feature which allows you to temporarily leave your terminal without having to exit any sessions which may provide unauthorized access to sensitive information. With the screen lock feature, Facet*Term* will ask for a password, then go into a "screen saver" mode. It will then require entry of the password to resume the Facet*Term* session. While you're away from your terminal, no one can access the applications you are running or see the contents of any windows you have active.

To activate the screen lock from the window command line, press

[CTRL][W] [X] [L] [Y]

The command line will prompt for the "lock word". It will accept up to 10 characters. It will then prompt for you to re-enter the lock word again. If you do not enter it the same both times, the operation will be terminated. After entering the same lock word twice, Facet*Term* will clear the screen and move the string "FacetTerm" around on the screen to let you know that the screen is locked. When you are ready to return to your session, press any key and Facet*Term* will prompt you to enter the lock word again. After you enter the lock word, your screen will be refreshed as it was before you locked it.

To activate the screen lock from the Facet*Term* menu, choose "Options" on the menu bar, then "Screen Lock" on the pull-down menu:

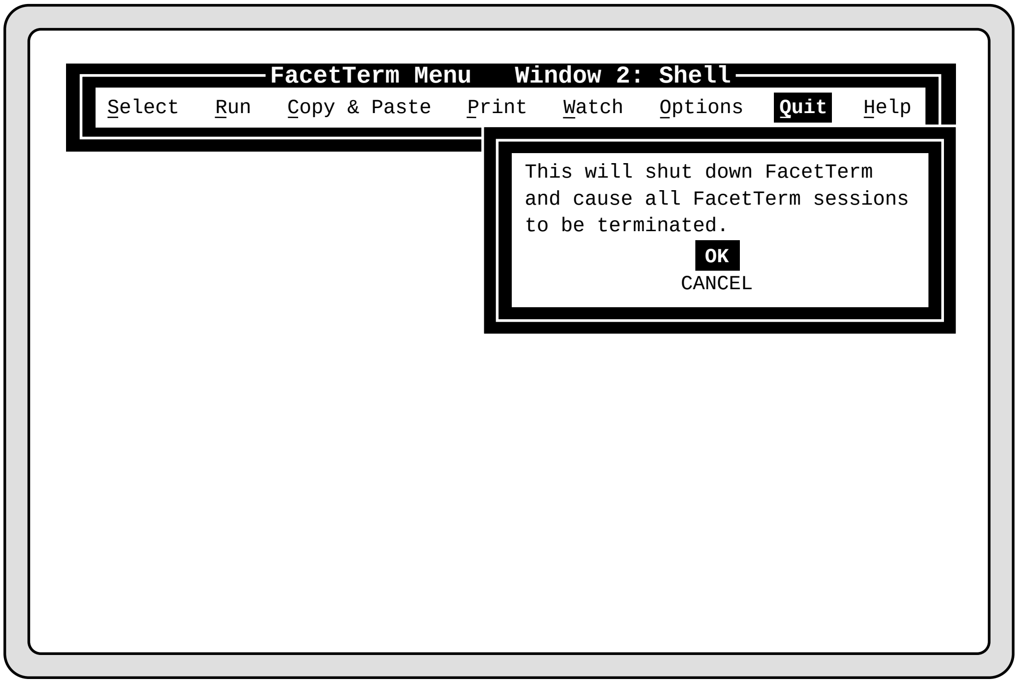

FacetTerm Menu   Window 1: Shell
Select   Run   Copy & Paste   Print   Watch   Options   Quit   Help
Keys...
Split screen...
Window title...   ^W'
Screen lock   ^Wxlv

The menu will then be removed and the window command line will come up and prompt for the lock word. From that point on, the procedure will be the same as if you had entered it from the window command line.

Quitting Facet*Term*

Using the Facet*Term* Menu to End a Session

You may exit from Facet*Term* by bringing up the menu and selecting "Quit" on the menu bar. An information box will be presented telling you that quitting Facet*Term* will cause all programs running in windows to be terminated. Select the "OK" item to terminate Facet*Term*:


FacetTerm Menu   Window 2: Shell
Select   Run   Copy & Paste   Print   Watch   Options   Quit   Help

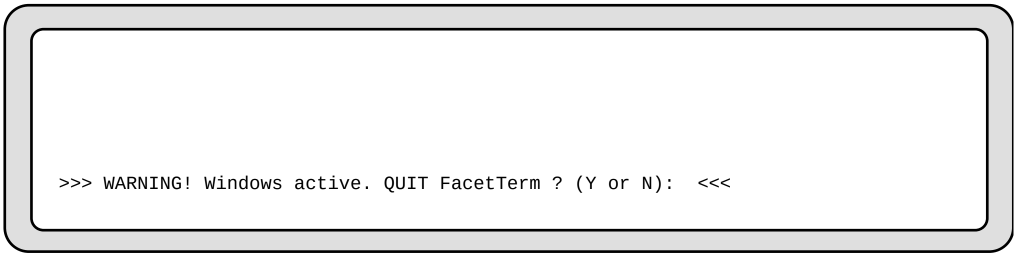
This will shut down FacetTerm
and cause all FacetTerm sessions
to be terminated.

Ending a Facet*Term* Session with the Window Command Line

You can also exit Facet*Term* from the window command line:

CTRL W Q

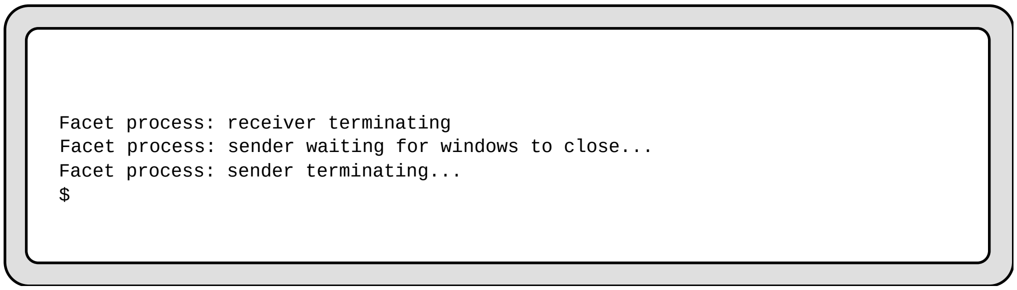
If there are still windows which are active, you will receive a warning and will be asked whether you still want to quit:



```
>>> WARNING! Windows active. QUIT FacetTerm ? (Y or N):  <<<
```

If you answer yes, then all programs running in active windows will be terminated and Facet*Term* will exit. If you answer no, the window command line will be removed and you will return to the current window. If no windows are active, you will also be asked if you want to quit, but there will be no warning about active windows.

Regardless of the method you use to quit Facet*Term*, as it shuts down, you will see the messages:



```
Facet process: receiver terminating
Facet process: sender waiting for windows to close...
Facet process: sender terminating...
$
```

There may be a delay after the "... waiting for windows to close..." message if the programs running in the windows do not respond to the hang-up signal. Any program which does not terminate on receipt of a hang-up signal will be sent a kill signal (which cannot be ignored).

Note: If you ever need to kill Facet*Term* from another terminal, do not use "kill -9". This will not give Facet*Term* the opportunity to shutdown programs running on windows. Instead, use a simple "kill" command or "kill -1".

Facet*Term* Configuration Guide

Facet*Term* has many features which may be customized. Most customization is done with the .facet file. Other files which can be customized to your own needs include the .facetlines file, the facettext file, the menu control files, function key files, the .facetalias file, and UNIX environment variables.

This chapter includes the following:

❏ An overview of the system administration menu

❏ A discussion of Facet*Term* terminal description files, and the .facetalias file

❏ An overview of Facet*Term* configuration methods

❏ Reference documentation of the .facet file

❏ Environment variables used by Facet*Term*

❏ Use of the .facetlines file for fixed terminal assignments of Facet*Term* devices

❏ An overview of the Facet*Term* text files

❏ A guide to customizing the menus

❏ Reference documentation for Window Watch files (.facetwatch.df)

❏ How to create function key files

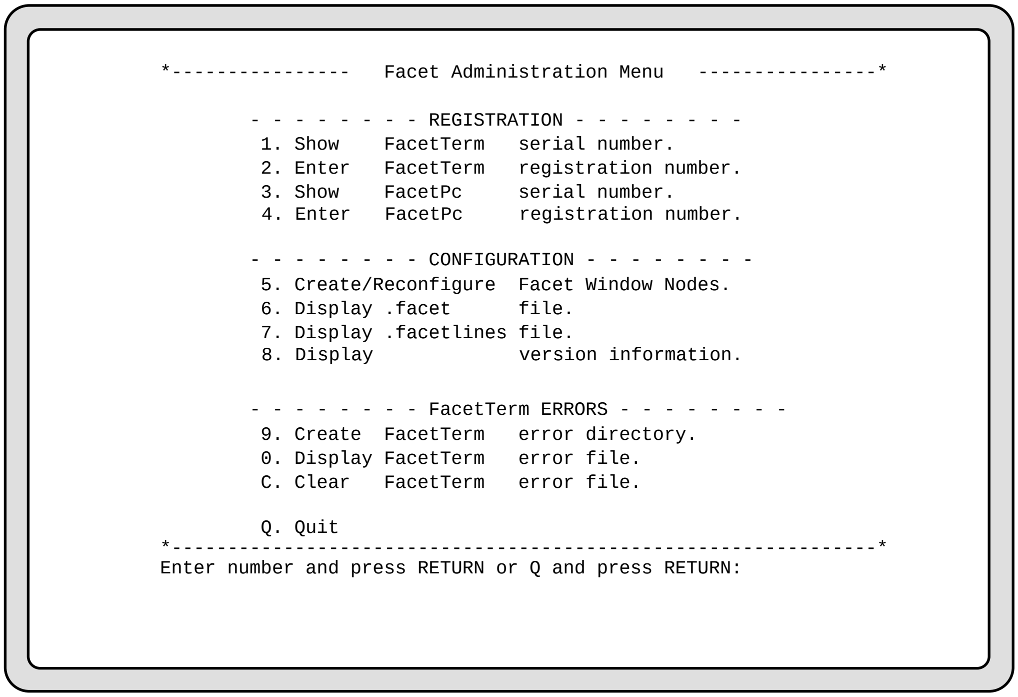
❏ Facet*Term* utility programs

The Facet Administration Menu

The Facet Administration Menu provides a convenient interface for registering Facet*Term*, viewing version and configuration information, configuring Facet*Term* window devices, and viewing Facet*Term* errors files. To start the administration menu, run the command:

facetadm

The program will present the following screen:



```
*----------------   Facet Administration Menu   ----------------*

        - - - - - - - - REGISTRATION - - - - - - - -
         1. Show    FacetTerm   serial number.
         2. Enter   FacetTerm   registration number.
         3. Show    FacetPc     serial number.
         4. Enter   FacetPc     registration number.

        - - - - - - - - CONFIGURATION - - - - - - - -
         5. Create/Reconfigure  Facet Window Nodes.
         6. Display .facet      file.
         7. Display .facetlines file.
         8. Display             version information.

        - - - - - - - - FacetTerm ERRORS - - - - - - - -
         9. Create  FacetTerm   error directory.
         0. Display FacetTerm   error file.
         C. Clear   FacetTerm   error file.

         Q. Quit
*---------------------------------------------------------------*
Enter number and press RETURN or Q and press RETURN:
```

You choose items from the administration menu by entering the number of the function you wish to perform followed by a Return. Each function will prompt you for additional information if needed. Many of these functions will only be entered at the request of a Facet*Corp* support engineer.

Registering Facet*Term*

Functions that you will almost certainly use are the serial number and registration items. If you choose the "Show Facet*Term* serial number" item, it will display your Facet*Term* serial number which must be supplied to Facet*Corp* in order to generate your registration number. It will also indicate whether your Facet*Term* is currently authorized, and for how many users.

Once you have your registration number from Facet*Corp*, choose the "Enter Facet*Term* registration number" item. It will prompt for the registration number which you have received. After you have entered the registration number, the serial number will be displayed again. Be sure that it indicates that your system is authorized for the proper number of users.

Creating Window Devices

If you have Facet*Term* for a system on which Facet*Term* installs a device driver, you may need to change the number of window devices which are configured. The installation instructions that came with your copy of Facet*Term* will indicate if your system is one for which this configuration is required.

Displaying Version Information

The "Display version information" selection on the menu will display all Facet*Term* version information available for your system. You should run this selection before calling for technical support. The PARTNUMBER, TERM and FACETTERM information reported will be particularly helpful to the support engineers.

Facet*Term* Terminal Descriptions

Facet*Term* terminal description files describe to Facet*Term* how to use a particular terminal in a particular emulation or mode. Many terminal features must be supported, and there are many interactions between terminal features. For this reason, ***the terminal descriptions are NOT considered to be user modifiable***. However, you might be asked by a support engineer to make a specific modification. This section contains some general information about how Facet*Term* uses the terminal description files.

Terminal Description Search Path

The complete search path for terminal description files is:

1. The directory in which Facet*Term* was started.
2. The $FACETINFO directory.
3. The $HOME directory.
4. The /usr/facetterm/localterm directory.
5. The /usr/facetterm/term directory.
6. The /usr/facetterm directory.

Alias Files

If the description is not found, an alias file is searched for. Alias files allow you to assign additional terminal names to a description, without having to copy the description file to a new name. The files .facetalias and .facetaliasM (for Machine dependent) are searched for along the same path as the terminal descriptions. When an alias file is found, Facet*Term* looks in the file for an alias file name which matches the name of the file it is searching for. The alias file consists of lines which contain:

1. The file name that Facet*Term* is looking for.
2. One or more spaces or tabs.
3. The name of the actual file that should be used.

Blank lines or lines beginning with "#" are treated as comments.

Default .facetalias and .facetaliasM files come with Facet*Term* and are installed in the /usr/facetterm/term directory.

Configuring Facet*Term*

Many of Facet*Term*'s features may be configured in a number of different ways. The .facet file is the most complete source for configuration. However, you may want to have users share a common .facet file, and yet have a few options different from one user to another. Therefore, there are other ways of configuring Facet*Term* which take precedence over what is specified in the .facet file. Some options are configurable via environment variables, and some options are configurable via arguments on the "facetterm" start-up command.

The precedence of these configuration options is:

1. Start-up arguments take precedence over environment variables and the .facet file.

.2 Environment variables take precedence over the .facet file, but are overridden by start-up arguments.

3. The .facet file options are overridden by the other configuration methods.

In the following sections you will find a reference to the .facet file and to Facet*Term*'s use of environment variables. Start-up arguments are documented in the User's Guide chapter under the section titled "Starting Facet*Term*".

.facet File Reference

This section includes a listing of the default .facet file which is installed in the /usr/facetterm directory. The file contains much internal documentation. In this section, we will divide the .facet file into topics and discuss each one.

Any line in a .facet file which begins with "#" is considered to be a comment. Every Facet*Term* feature available from the .facet file is included in the default file, but with most of the features commented out. To enable a feature, simply remove the "# " and any leading spaces from the beginning of the line.

The Default .facet File

```
#############################################################################
# FacetTerm user customizable file:  .facet
# used for customizing FacetTerm behavior
#
# Please copy this file into your home directory for customization.
# FacetTerm will search your home directory for this file at startup, if
# its not found FacetTerm will use the default file (.facet) in /usr/facetterm.
# lines that begin with # are treated as comments
#############################################################################
```

The default .facet file is installed in the /usr/facetterm directory. If you are installing over a previous version of Facet*Term*, this file will not have overwritten your original .facet and will be named .facet.df (.df for default). You may copy this file to users' $HOME directories and modify it individually for each user.

Specifying What Programs to Start Automatically

```
### Starting Applications:
###     To run a program on a window, specify the window number, a tab or a
###     space, and the program name.   “-sh” will run .profile “sh” will not.
###     The tenth window is window number 0.
###     To specify a title for a window, specify the title after
###     “window_title=”.   The title applies to immediately following line.
###     The title is used in the User Interface select menu and sometimes
###     on softkeys.
###     The word respawn may be placed in front of the window
###     number in the “program name line” to restart the application if it
###     ever gets killed while FacetTerm is running.
###     To specify the “last” window, use the capital letter “L”.   The user
###     is responsible for insuring that no window has multiple programs
###     run on it.

window_title=Shell
1 -${SHELL:-/bin/sh}
# respawn 1 -${SHELL:-/bin/sh}

window_title=Shell
2 -${SHELL:-/bin/sh}

# window_title=WINDOW 3 PROGRAM
# 3 program

# window_title=WINDOW 4 PROGRAM
# 4 program

# window_title=WINDOW 5 PROGRAM
# 5 program

# window_title=WINDOW 6 PROGRAM
# 6 program

# window_title=WINDOW 7 PROGRAM
# 7 program

# window_title=WINDOW 8 PROGRAM
# 8 program
```

Configuring a Printer Attached to Your Terminal as an LP Printer

```
### Local lp setup, for if you have an lp printer hooked to your terminal.
###     Sets the window to “print mode” and reads from a named pipe.
# window_title=PRINTER CONTROL
# 9 ${FACETTERMPATH:-/usr/facetterm}/bin/fct_lpwin
```

If your terminal supports an attached printer with transparent print capabilities, you may use a window to control the printer as a system spooler printer for use with the UNIX lp commands. The example above will use window 9 to run the fct_lpwin program which will control the printer attached to your terminal. Note that if you limit Facet*Term* to fewer than 9 windows, you must start this program on a different window.

Before you can use the lp commands to send print jobs to this printer, you must add the printer to the lp configuration. This is done with the command (which you will run from a shell - not from the .facet file):

fct_lpadmin -a *ttyname* **-m** *lp_model_name*

where *ttyname* is the name of your physical tty line (not the name of the window device). You can determine this name by running the “tty” command before starting Facet*Term*. *lp_model_name* is the name of the printer model that should be used to control the type of printer attached to your terminal. Consult your system administrator or your system’s documentation on the “lp” commands for help with this name. On most systems, you must be logged in as root or administrator to run this command.

To remove the attached printer from the lp configuration, use the command:

fct_lpadmin -d *ttyname*

where *ttyname* is the name of your physical tty line.

To print to the attached printer, you will use the standard lp command, and specify the destination to be the attached printer:

lp -d fct_*ttyname*

where *ttyname* is the name of your physical tty line. The "fct" prefix is attached by the fct_lpadmin command. For example, if your physical tty line is tty01, then your attached printer destination name will be fct_tty01.

Automatically Starting the Facet*Term* Menu

```
### FacetTerm User Interface menu program, accessed by "menu_hot-key"(see below)
window_title=FacetTerm Menu
L ${FACETTERMPATH:-/usr/facetterm}/bin/fct_ui
```

The Facet*Term* menu is automatically started in the last available window with the default .facet file.

Specifying the Menu Hot-Key

```
### The default User Interface hotkey is "Control-F".  To change this to
###     another character specify the character after "menu_hotkey=".
###     When this entry is present, the hot key is displayed on startup
###     and "\m" is replaced by the hot key in function key files.
###     Comment out this entry if you do not use the User Interface.
menu_hotkey=^F
```

The menu hot-key will default to

CTRL F

even if you do not make this specification in the .facet file. However, this specification is necessary in order for Facet*Term* to print the hot-key on the startup screen. If you are not using the menu, you will want to comment out the hot-key specification. The hot-key may be changed by specifying another key in the "menu_hotkey=" statement.

Specifying an Alternate Window Command Line Hot-Key

```
################################################################################
### Below are user preference FacetTerm options that may be customized

### The default FacetTerm command line hotkey is "Control-W".
### To change this to another character, specify the character after "hotkey=".
# hotkey=^W
```

The Facet*Term* window command line hot-key will be

CTRL W

unless it is specified to be something different here.

Limiting the number of windows that Facet*Term* allocates

```
### To limit the number of windows that FacetTerm allocates, specify
###     “windows=” followed by the desired number of windows.
###     This is equivalent to specifying the number of windows on the
###     FacetTerm command line.
# windows=10
```

By default, Facet*Term* will allocate enough memory to run 10 windows. If you want to limit this memory utilization, you can restrict the maximum number of windows that Facet*Term* will run either by specifying it on the Facet*Term* startup command line, or here in the .facet file.

Automatically activating a Window Watch file when Facet*Term* starts

```
### To activate a “WindowWatch” file when FacetTerm starts, specify
###     “windowwatch=” followed by the desired file name.
###     If no filename is specified, the default name .facetwatch is
###     used. (You must create this file.)
###     For example files, see /usr/facetterm/text/*.fw and
###     /usr/facetterm/text/.facetwatch.df
###     These files are searched for along the same path as the
###     FacetTerm text files.
# windowwatch=
# windowwatch=mailum.fw
# windowwatch=mailusm.fw
###     If the “WindowWatch” file that you are activating is specific
###     to a window, place a dash and the window number after
###     window_watch and before the equal sign.
# windowwatch-2=
# windowwatch-2=actived.fw
# windowwatch-2=activepatd.fw
# windowwatch-2=activepatw.fw
# windowwatch-2=activew.fw
```

If you want to activate the same window watch file(s) each time you start Facet*Term*, you can do so in the .facet file rather than manually having to activate the file from the menu or window command line. The format of a window watch file is described later in this chapter in the section titled "Facet*Term* Watch Files (Window Watch)."

Specifying a Default Function Key File

```
### The default function key definitions for a terminal can be changed by
###     specifying the desired contents of each key in a function key file.
###     A supplied function key file that loads the function keys with
###     window selection commands, “WINDOWS.ftkey”, can be loaded by
###     specifying “function_keys=” without a filename.   To load any other
###     function key file,  specify its filename after “function_keys=”.
# function_keys=
# function_keys=lab82_8.ftkey
# function_keys=lab8_8.ftkey
# function_keys=sl_16.ftkey
# function_keys=ti8_8.ftkey
# function_keys=WINDOWS.ftkey
# function_keys=SWINDOWS.ftkey
###     To load only one window, place a dash and the window number after
###     function_key and before the equal sign.
# function_keys-3=test.ftkey

### To cause the function keys to be programmed to the terminals’s default
###     values as FacetTerm starts, specify “no_assume_default_function_keys”.
# no_assume_default_function_keys
```

Rather than having to load function key files from the menu or window command line, they can be automatically loaded at startup by making the specification in the .facet file. The comments in the default .facet file fully explain this feature.

Quitting When All Windows are Idle

```
### To automatically exit FacetTerm when all of the programs on all of the
###     windows have stopped,  specify "quit_on_all_windows_idle".
# quit_on_all_windows_idle
```

Quitting Without a Confirmation Prompt

```
### To prevent FacetTerm from prompting for confirmation before exiting on
###     a "Control-W  Q" command,  specify "no_quit_prompt".
# no_quit_prompt
```

Preventing User from quitting FacetTerm while windows are still active

```
### To insist that all applications be shut down before exiting FacetTerm
###     specify "diable_quit_while_windows_active".
###    Note that this does not prevent a program ( eg. fct_command  fct_ui )
###    from requesting a quit if it is the only active window.
# disable_quit_while_windows_active
```

Clearing Windows When They Become Active or Become Idle

```
### When "clear_window_on_open" is enabled,  an automatic clear of the window
###     is done when a window goes from idle to active.
# clear_window_on_open

### When "clear_window_on_close" is enabled,  an automatic clear of the window
###     is done when a window goes from active to idle.   Note that this may
###     have the effect of erasing a error message explaining why a program
###     would not run.
# clear_window_on_close
```

Enabling the Facet*Term* Screen Saver

```
### To enable the "screen saver" function, specify "screen_saver=" followed
###     by the number of seconds.  Useful for some terminals.
###     To change the default 5 seconds between moving the text,
###     specify "screen_saver_interval=" followed by the number of seconds.
###     To change the default text string, specify "screen_saver_text="
###     followed by the desired text.
###     ( You can invoke the screen saver in window command mode with ^Wxs ).
# screen_saver_timer=300
# screen_saver_text=FacetTerm - press any key to continue
# screen_saver_interval=5
```

If your terminal does not have a screen saver (or if you are running Facet*Term* on the memory mapped console of a computer) you may want to use the Facet*Term* screen saver feature. This will blank the screen after a time-out interval and move a message around on the screen. The message will indicate that the screen saver is on, and that pressing any key will refresh the screen. The comments in the default .facet file fully explain how to set up this feature.

Changing the Default Type of Copy

```
### To change the default type of copy for copy and paste, specify one of the
###     following.
### "block_copy" takes characters in a rectangle from the top cursor to
###     the bottom cursor.
###     Trailing blanks are eliminated and multiple lines are separated by
###     "return".
### "vertical_copy" takes the same characters as "block_copy".
###     No "return's are added.  Trailing blanks are collapsed to a single blank.
### "stream_copy" takes characters from the top cursor to the right margin,
###     all lines between the top and bottom cursor,
###     and the left margin to the bottom cursor.
###     Trailing blanks are eliminated and multiple lines are separated by
###     "return".
### "wrap_copy" takes the same characters as "stream_copy".
###     No "return"s are added.   Trailing blanks on a line and leading blanks
###     on the next line, if one of both exist, are collapsed to a single blank.
### You can change the type of copy in command mode by specifying "s", "b", or
###     "w" while a copy is active.
# block_copy
# vertical_copy
# stream_copy
# wrap_copy
```

The default type of copy is block copy. The comments in the default .facet file fully explain how to change this default.

Allowing a "Beep" on an Off-screen Window to Sound

```
### Windows specified as allow_beep_offscreen can cause the terminal to
###     beep even if they are not on the screen.   This can be specified for
###     as many windows as desired.
# allow_beep_offscreen=9
```

The ASCII ^G character is interpreted by most terminals by "beeping". By default, Facet*Term* will not sound the "beep" when a ^G is received on a window which is not the current window. Since ^G is not a printable character, it will not be refreshed when you switch back to the window. If you would like the beep to be sounded even when the window is off-screen, you may specify this feature as described in the comments in the default .facet file.

Blocking Output on a Non-current Window

```
### To indicate that characters should not be sent to a window unless
###     the window is current, specify "window_blocked=" followed by the
###     window number.
###     Any number of windows can be so specified.
# window_blocked=2
```

Normally, Facet*Term* will continue accepting output from programs that are on off-screen windows and updates its image of the window. If you do not want output to continue on an off-screen window, you may make the specification shown above. Note that this will probably cause the program running on the window to stop as its tty buffer fills up. Then when the window is selected and its output is read, it will continue.

Locking Window 1 in a Page on a Terminal With Screen Pages

```
### When “lock_window_1” is enabled and you are running on a multi-page
###     terminal, the screen for window 1 will remain in the terminal
###     whenever possible.
# lock_window_1
```

Specifying Transparent Print Options

```
##############################################################################
### Below are FacetTerm options or parameters less likely to need customization

### To cause a window to automatically be in transparent print mode,
###     specify "transparent_print_window=" followed by the window number.
###     To override the default transparent print operation, specify
###     "transparent_print_buffer_size=" followed by a number of characters
###     to be sent in a single transparent print sequence.
###     To set a minimum time for no output from other windows before
###     transparent print operations are started, specify
###     "transparent_print_quiet_timer=" followed by a number of milliseconds.
###     To set a minimum time for no output from other windows before
###     continuing transparent print operations, specify
###     "transparent_print_idle_timer=" followed by a number of milliseconds.
###     To set a maximum time to which FacetTerm may automatically increase the
###     transparent_print_idle_timer when encountering xoff delays,
###     specify "transparent_print_idle_timer_max" followed by a number of
###     milliseconds.
###     To set a minimum time after an xoff delay to the terminal before
###     continuing transparent print operations, specify
###     "transparent_print_full_timer=" followed by a number of milliseconds.
###     To set how closely one second delays must appear to count as an
###     xoff delay, specify "transparent_print_delay_count" followed by a
###     number.
# transparent_print_window=9
# transparent_print_buffer_size=64
# transparent_print_quiet_timer=2000
# transparent_print_idle_timer=100
# transparent_print_idle_timer_max=1000
# transparent_print_full_timer=10000
# transparent_print_delay_count=5
```

These parameters are all available to allow you to tune your transparent print performance to the speed of your printer, terminal, and tty line. The comments in the default .facet file are fairly complete, but you may require the help of a support engineer if you feel that you need to make these adjustments.

Specifying the Handling of the Break Key

```
### To indicate the handling of the break key, specify "send_break_to_window"
###     or "send_break_to_facetterm".    Specify "use_PARMRK_for_break" to
###     cause the termio "parmrk" setting for break handling or
###     "use_NULL_for_break" for settings where "break" is received as a
###     "null" and "null" is treated as "break".
# send_break_to_window
# send_break_to_facetterm
# use_PARMRK_for_break
# use_NULL_for_break
```

By default, Facet*Term* will bring up the window command line when a break is received. You may specify that the break be sent on through to the application running in the window. In addition, there are statements which allow you to specify how the break should be indicated to the application running in the window.

Disabling the Window Command Line Run Command

```
### To prevent users from running programs with the "Control-W  R" command,
###     specify "disable_control_W_r".
# disable_control_W_r
```

This directive is useful in cases where it is desired to restrict the programs that a user may run to those things that are started from his .facet file or which are on his Run menu.

Specifying Scroll Memory Parameters for Terminals With Scroll Memory (HP)

```
### To override the default number of lines of scroll memory retained on
###     terminals with scroll memory, specify "scroll_memory_lines=" followed
###     by the desired number of lines.
###     To override the default baud rate at which lines of scroll memory
###     are retained specify "min_scroll_memory_baud=" followed by the
###     baud rate.
# scroll_memory_lines=25
# min_scroll_memory_baud=9600
```

These directives are for use with HP terminals which have scrolling memory. Facet*Term* will remember as many lines scrolled off the top of the screen that you specify. When you switch windows, the scroll memory will need to be refreshed as well as the current contents of the visible screen. Therefore, you would probably not want to specify a large amount of scroll memory. You should also be aware that any off-line operations you perform with the terminal on the scroll memory contents will not be available to Facet*Term*.

HP Transparent Print of Binary Data

```
### With FacetTerm, HP terminals can pass 8 bit binary transparent print data
###     in HP mode but not in ansi mode.   If both modes are to be used and 8 bit
###     binary data is to be sent, specify "print_only_hp_personality" to
###     suspend transparent print while in ansi mode.
# print_only_hp_personality
```

If you have an HP terminal with an HP LaserJet attached, you may have a need to send 8 bit data to the printer. The HP terminals will only support 8 bit printer data when in HP mode. This directive tells Facet*Term* to only do transparent print when the terminal is in HP mode rather than ANSI mode.

Specifying the Idle Window and its Parameters

```
### When "idle_window" is set to a window number and the current window
###     goes from active to idle,  an automatic window select is done to the
###     window specified as the idle_window.
###     If the "idle_window" is specified as "A", the next active window is
###     selected rather than a specific window.
###     See also the "Control-W  I" command.
# idle_window=0
# idle_window=A

### To paste a specified string to the idle window rather than selecting it,
###    specify the desired characters after "idle_window_paste_chars=".
# idle_window_paste_chars=^Z

### When the window specified as the "idle_window" goes from active to idle,
###     the "idle_window" is automatically set to "none".   When
###     "idle_window_no_cancel" is enabled, the "idle_window" is not changed.
# idle_window_no_cancel
```

You may designate a window to be the idle window, meaning that it is selected or sent a string of characters when any window goes idle. This feature is primarily for use by the Facet*Term* menu and is set up programmatically by the menu. However, you may set up the feature in the .facet file as described in the comments.

Specifying the Windows Window and its Parameters

```
### When "windows_window" is set to a window number and FacetTerm receives a
###     break or a null,  the "windows_window_char_start" is sent to that
###     window.   See also the  "Control-W  W"  command.
# windows_window=0

### When the window specified as the "windows_window" goes from active to idle,
###     the "windows_window" is automatically set to "none".   When
###     "windows_window_no_cancel" is enabled, the "windows_window" is not
###     changed.
# windows_window_no_cancel

### The character sent the "windows_window" when a break or a null is pressed
###     defaults to "Control-Z".   This can be changed by specifying the
###     desired character in "windows_window_char_start".
# windows_window_char_start=^Z

### When a program in a window invokes the "pop-up" command, the character
###     sent to it when leaving the "pop-up" mode defaults to "Control-B".
###     This can be changed by specifying the desired character in
###     "windows_window_char_stop".
# windows_window_char_stop=^B

### The window command  "Control-W  X  N  Y"  allows a program to request that
###     it be notified when it is the current window.   The notification
###     character defaults to "Control-Y".   This can be changed by specifying
###     the desired character in "notify_when_current_char".
# notify_when_current_char=^Y

### When commands are sent to FacetTerm from a program ( using a facility
###     such as the "fct_command" program ), the command is usually canceled
###     if it ends prematurely.   However, if the command ends with "Control-C"
###     FacetTerm will prompt the user for the remainder of the command.
###     This default character can be changed by specifying the desired
###     character in "window_mode_to_user_char".
# window_mode_to_user_char=^C
```

The windows window is also a special feature that is primarily used by Facet*Term* aware programs, but which may be set up as described in the comments.

Specifying the Default Directory and Parameters for Capture Files

```
### The default directory for writing capture files is the current working
###     directory where FacetTerm was started.   To change this default
###     directory specify the desired directory after "capture_directory=".
# capture_directory=/usr/facetterm/capture

### By default, FacetTerm uses the normal buffering in the "stdio" package
###     when writing capture files.   To cause the captured characters to
###     be written to the file without buffering, specify "capture_no_buffer".
# capture_no_buffer
```

Specifying Split Screen Options

```
### To prevent FacetTerm from displaying the window number on the split screen
###     dividers, specify "no_split_numbers".
# no_split_numbers

### To prevent FacetTerm from displaying the split screen dividers ( i.e.
###     leave them blank ), specify "no_split_dividers".
# no_split_dividers
```

Altering the Facet*Term* Shutdown Signalling Behavior

```
### When FacetTerm shuts down, it sends a hangup to the windows and waits
###     20 seconds before sending a kill signal.   To change this time,
###     specify "kill_seconds=" and the desired number of seconds.
# kill_seconds=20

### To prevent FacetTerm from sending the kill signal to windows, specify
###     "no_kill_processes_on_shutdown".
# no_kill_processes_on_shutdown

### To enable Facetterm to send the kill signal to windows on systems
###     that default to not sending the signal, specify
###     "kill_processes_on_shutdown".
# kill_processes_on_shutdown
```

Altering the Facet*Term* Outlining Behavior While in the Copy Command

```
### When FacetTerm is doing a screen copy, it will move the cursor from corner
###     to corner of the copy every half second.   To change this timing,
###     specify "copy_and_paste_timer=" and the number of half_seconds.
###     0 means do not move the cursor.
# copy_and_paste_timer=1
```

Alter Facet*Term*'s Behavior When it Receives an unrecognized Escape Sequence

```
### Override the default action to be taken when a sequence is not recognized.

# error_ignore
# error_pass
# error_control_ignore
# error_control_pass
```

When Facet*Term* receives an escape sequence or control character from an application that it does not understand, it normally "beeps" and prints the sequence on the screen. These directives modify that behavior. The "error_ignore" directive causes Facet*Term* to discard escape sequences and control characters that it doesn't understand. The "error_pass" directive causes Facet*Term* to simply pass escape sequences and control characters it doesn't understand on through to the terminal. The "error_control_ignore" directive causes Facet*Term* to discard control characters that it doesn't understand (but continue to "beep" unrecognized sequences). The "error_control_pass" directive causes Facet*Term* to pass control characters that it doesn't understand on through to the terminal. Note: Because the use of these directives masks errors, they should only be used at the request of a Facet*Corp* support engineer.

The .facetprint Script

When Facet*Term* is installed, a default .facetprint file is installed in the /usr/facetterm directory. The .facetprint file is a script which is called by Facet*Term* when you do a screen print. You may customize .facetprint scripts and control which script users access by the directory in which they are placed. Facet*Term* looks for the .facetprint script in the following order:

1. The directory in which Facet*Term* was started.
2. The home directory indicated by the UNIX environment variable $HOME.
3. The /usr/facetterm directory.
4. The /usr/facet directory.

This allows the method of printing to be specified by application, by user, or for the whole system.

If the .facetprint script cannot be found, or is not executable, Facet*Term* will beep to indicate that it could not execute the command.

The .facetprint script is called with two arguments which are the names of temporary files which hold the screen contents. Only the characters of the screen are placed in the first temporary file; any special screen attributes (such as underlining or alternate character sets) are ignored. Each line of characters has trailing blanks trimmed off and is followed by a return character. You can program a simple .facetprint file to simply call the UNIX lp command with this first file name:

```
# a very simple .facetprint script
lp  -c  -s  $1
```

The file named by the second argument to the .facetprint script contains not only the screen characters, but also information about character sets and attributes. The default .facetprint script uses this second file and sends it as the standard input to the "fct_printx" program for processing of the special character set and attribute information. The output of fct_printx can then be piped into "lp" or whatever you want.

The fct_printx program gets information from your terminal description file and a printer description file in order to properly map special characters and attributes from the terminal to the printer. It uses the $FACETTERM or $TERM environment variables to determine which terminal description file to read, just as the "facetterm" program does. It uses the first argument passed to it or the $FACETPRINTER environment variable to determine which printer description to use. If fct_printx does not have enough information to do the mapping, it will only process the characters and ignore the character set and attribute information.

Facet*Term* deletes the temporary files after the .facetprint script is run.   If your system's print command does not create a link to the file, copy the file, or rename the file, then the .facetprint script must do so, and delete the copy after it is printed.

Environment Variables Used by Facet*Term*

Facet*Term* will use the value of various UNIX environment variables to alter its behavior. The following is a list of each variable and its effect on Facet*Term*.

Variables That Affect Search Paths

FACETCAPTUREDIR

Used to specify the directory where capture files should be placed. For example, adding the following lines to a user's .profile:

```
FACETCAPTUREDIR=/usr/john/capture
export FACETCAPTUREDIR
```

will cause all capture files from his Facet*Term* sessions to be written to the /usr/john/capture directory.

FACETINFO

Used to specify the first directory searched for Facet*Term* terminal description files.

FACETTEXT

Used to specify the first directory searched for Facet*Term* text files.

FACETUTMPDIR

Used to specify an alternate directory for locating the utmp file to be updated with entries for each active window. Normally, Facet*Term* updates the system's /etc/utmp file with an entry for each active window. This has the effect of showing all Facet*Term* windows being used on the system when you run the "who" command. If you do not want Facet*Term* to update the system utmp file, you can set this variable to point to another directory. For example:

```
FACETUTMPDIR=/tmp;  export FACETUTMPDIR
```

would cause Facet*Term* to write to a utmp file in the /tmp directory, and then the "who" command would only show login sessions on terminals and not Facet*Term* sessions on each window. Note that some software requires that the device it is running on have an entry in the system utmp.

HOME

The users home directory, as determined by this variable, is used in the Facet*Term* search paths.

LANG

This variable, if set, specifies the language to be used. This value is used in the Facet*Term* text search path.

Variables That Affect Hot-Keys

FACETHOTKEY

Used to change the Facet*Term* hot-key from its default value of "^W". The following lines in a user's .profile:

```
FACETHOTKEY='^X'
export FACETHOTKEY
```

will cause the Facet*Term* hot-key to be "^X" next time Facet*Term* is started. Note, that if you change this variable and you are currently running Facet*Term*, you must exit Facet*Term* and restart it in order to get the new hot-key. This environment variable will override the "hotkey=" in the .facet file, but not the "hotkey=" on the command line.

FACETNOPROMPTHOTKEY

This environment variable can be used to specify a second hot-key which will access the window command line. This hot-key differs from the normal window command line hot-key in that it causes the window command line to accept commands, but not write any prompts on the screen. This feature is useful when mapping a key to perform a series of window commands, and you don't want to have the window command line prompts flashing by at the bottom of the screen. For example, you could define the no-prompt hot-key as "^Y" with:

```
FACETNOPROMPTHOTKEY='^Y'
export FACETNOPROMPTHOTKEY
```

You could then run Facet*Term* and use the window command line to map "^N" to perform a "Next window" funtion with the command to the window command line:

CTRL W : **^N^Y+\s**

which means: map "^N" to be "^Y" followed by "+" then a space. Now when you press "^N" it is converted to the no-prompt hot-key ("^Y"), which accesses the window command line, and then the window command line interprets the "+" as "go to the next active window", and the space as "cancel the window command line". The net result is that "^N" now causes the next active window to be selected with no window command line prompts flashing by on the bottom of the screen.

FACETMENUHOTKEY

Used to change the Facet*Term* pull-down menu hot-key from its default value of "^F".

Variables That Affect Which Terminal Description is Used

FACETTERM

Used to specify an alternate terminal description file. You may set this variable, and Facet*Term* will use the value of the variable as the name of an alternate terminal description. It will look along the terminal description search path for the file named $FACETTERM.fi. If you specify an alternate terminal description on the command line when starting Facet*Term*, the FACETTERM variable will be set to the specified value.

TERM

If no alternate terminal description is specified with the FACETTERM variable or a command line argument, then $TERM.fi is used as the name of the terminal description file to use.

FUNCTION_KEYS

If this variable is set to NO, such as:

FUNCTION_KEYS=NO; export FUNCTION_KEYS

then Facet*Term* will not initialize, remember, or re-initialize individual function keys. Sequences that are recognized as being function key sequences are passed on to the terminal.

SHELL

This variable determines which shell program will be started with a

CTRL W S

command. For example,

SHELL=/bin/sh; export SHELL

would cause a Bourne shell to be started when you start a shell with Facet*Term.*

FACETPRINTER

This variable is used by the fct_printx program to determine which printer description to use.

FACETSHELL

This variable is used for the same purpose as the SHELL variable, but overrides it.

FACETUSER

Facet*Term* puts this variable into the environment, using the results of a call to cuserid. This is used to set the user id in /etc/utmp for the sessions.

The .facetlines File (Fixed Assignments)

Note: This feature is only available on systems where a Facet*Term* device driver has been added. If you wish to use this feature, but you are not sure whether the feature is available for your system, call for technical support.

Normally, Facet*Term* chooses the lowest numbered Facet line (and its associated set of windows) that is not in use. This default method has the advantage of automatically sharing the available lines and requires no management.

This method does mean, however, that the set of windows that will be chosen for a particular user is not predictable. That is, the same user on the same terminal may run on /dev/t8w1 through /dev/t8w10 one time, and on /dev/t10w1 through /dev/t10w10 the next time.

If your applications maintain tables that assign printers, access privileges, or terminal types based on the ttyname rather than the logname or environment, they will not work properly with this unpredictable default method of assigning lines. To use such applications with Facet*Term*, you need to assign a particular terminal to a particular set of windows. This is done by creating a file called .facetlines.

In addition to assigning a particular terminal to a particular set of windows, the .facetlines file allows you to specify another name for the assigned set of windows. This is useful for applications or utilities that base their behavior on the ttyname of the device on which they are running.

When Facet*Term* starts, it looks for the .facetlines file in:

1. The /etc directory.
2. The /usr/facet directory.

If you do not want users to be able to change this file, put it in the / etc directory and write-protect it appropriately. Otherwise, put it in the /usr/facet directory.

The .facetlines file consists of lines having the following format:

1. The terminal device name (as reported by the tty command).

2. One or more blanks or tabs.

3. The desired Facet line (starting with 0).

    Optionally:

4. One or more blanks or tabs.

5. The desired first part of the window names (which must start with /dev/ and end with w).

***Note:** Trailing spaces are not allowed in the .facetlines file.*

For example, the file:

```
#this is a comment
/dev/tty04      4
/dev/tty06      6       /dev/tty06w
/dev/ttya       1
```

will assign tty04 to Facet line 4, tty06 to Facet line 6, and ttya to Facet line 1. When Facet*Term* is run on any other terminal, it will not choose Facet lines 4, 6, and 1.

In addition, the window devices for Facet line 6 will be renamed from /dev/t6w1 ... /dev/t6w10 to /dev/tty06w1 ... /dev/tty06w10.

After editing the .facetlines file, you must run the command:

```
/usr/facet/facet.makenode
```

in order to have any device name changes made in the .facetlines file take effect. You must be logged in as root to run this command, and no users should be running Facet*Term* when you are running the facet.makenode command.

Facet*Term* Text Files

There is a group of files which we will refer to as Facet*Term* text files. These files include the menu control files which determine the structure of the menu; watch files which control the Window Watch feature; the facettext file which may be used to modify Facet*Term* messages; and facetuitext which may be used to modify the menu's messages (ui for user interface).

The Facet*Term* text files are searched for along a path which allows you to customize Facet*Term* for each user, for various languages, or for a group of users. Facet*Term* looks for its text files in the following directories in the order listed:

1. ./ (The directory in which Facet*Term* was started).
2. $FACETTEXT/$LANG/
3. $FACETTEXT/
4. $HOME/
5. ./.facettext/$LANG/
6. ./.facettext/
7. $HOME/.facettext/$LANG
8. $HOME/.facettext/
9. /usr/facetterm/localtext/$LANG
10. /usr/facetterm/localtext/
11. /usr/facetterm
12. /usr/facetterm/text/$LANG
13. /usr/facetterm/text (this is where the ".ex" default example files and the standard menu control files reside.)
14. /usr/facet/text/$LANG
15. /usr/facet/text

The $LANG environment variable is meant to be used to control the language used in a multi-lingual environment. In a common language environment, you would probably not want to use this variable.

The /usr/facetterm/localtext directory is meant as a place for you to make your own system-wide modifications that won't be overwritten in the future by Facet*Term* updates.

Customizing the Facet*Term* Menus

You can customize all of the Facet*Term* menus pull-down menus. The format of these files is described below. The default menu control files reside in the /usr/facetterm/text directory. The easiest way to create your own menu control files is to copy the default files to the desired directory (refer to the search path documented in the section above titled "Facet*Term* Text Files") and then modify them. You may copy and modify only selected files, and the default files will be used for those not found in the directory with your modified menu control files.

Menu Control File Format

Each menu control file describes a level of the complete menu structure. For instance, the menu bar is described in one menu control file, the "Run" pull-down menu is described in another menu control file, and so on. The top level menu for the Facet*Term* menu must be named "fterm_menu".

Each menu control file consists of groups of lines, with each group describing items in the menu. In addition, some lines describe general features of a particular menu level.

Each line begins with a key word which identifies what aspect of a menu or menu item is being described. The following is a list of the different types of menu control file specifications. Blank lines, and lines beginning with "#" are considered to be comments.

Menu Directives

menu_bar *row col*

Specifies that the menu is a menu bar type. If *row* and *col* are specified, the upper left corner of the menu is located at that position. Otherwise, the menu is located at a default position.

pull_down *row col*

Specifies that the menu is a pull-down type. If *row* and *col* are specified, the upper left corner of the menu is located at that position. Otherwise, the menu is located in a default position. A cascading menu which is entered from a pull-down menu is also specified as a pull_down menu. In addition, the top level menu may be a pull-down menu.

menu_title *xxx*

The remainder of the line after "menu_title" is used as the title for the menu. The title of the menu appears in a bar above the menu choices. If a menu title specification does not appear in the control file, no title bar will be drawn. Titles may include variables which are resolved at run time by the menu. The variables available are listed in a section below.

item_name *xxx*

An "item_name" line begins the definition of a new item in the menu. The remainder of the line after "item_name" is used as the name of the item in the menu. All following item specifications will apply to the current item started with the previous "item_name" entry.

item_selection *c*

The character following "item_selection" is used as the item selection character. The upper or lower case of the character will both work as an item selection character. The first instance of the item selection character in the item name will be underlined when the menu is presented (or highlighted in some other way if the terminal does not support underline). If you do not want the first occurrence of the character to be the one underlined, indicate the character you want underlined by preceding it with an "&" character in the item name string.

item_type *type_name*

Identifies the type of the item. Valid item types are:

program

Indicates that the item action will be a UNIX program which should be run in the current window if idle, or the next available window otherwise. The item action string is passed to the system() function call. Therefore, shell processing of the item action string is available.

menu

Indicates that the item action is a menu control file. Menu control files are searched for along the path described in the section above on the Facet*Term* text file search path.

intrinsic

Indicates that the item action is a keyword for a function which is intrinsic to the menu. The available intrinsic item actions are documented later in this section.

ft_command

Indicates that the item action is a string which should be sent to Facet*Term* as if the command line hot-key had been pressed. For example, an item action of

t1

would cause window 1 to be put in the top half of a split screen and be made the current window.

ft_cmd_to_user

This item action is like ft_command except that after the item action string has been processed by Facet*Term*, it will begin accepting input from the user. For example, an item action of

xc

would bring up the Facet*Term* command line, to the point of asking whether capture should be turned on or off and would allow the user to continue interaction with the command line.

item_action *xxx*

The remainder of the line after "item_action" is used as the action which is interpreted according to the action type as described above.

item_action_2 *xxx*

Same as item_action except that it is a second action to take. This may only be used with a fct_command item type. It is used to send two successive commands to Facet*Term*.

new_window_title *xxx*

The remainder of the line after "new_window_title" is used as the title of a new window which is activated as a result of the action. This is only used if the item type is a program since this is the only type of action which activates a new window. If no "new_window_title" specification is made then the item_name value is used as the new window's title. Window title specifications may include variables.

item_auto_select

Indicates that this item should be automatically selected when this menu is run. If multiple items claim to be the auto selection item, the last one specified is the one used.

dialog_box *row col*

Specifies that the current item, when selected will put up a dialog box to get some information from the user. The row and column for the upper left corner of the dialog box on the screen may be specified. Otherwise it will be placed in a default position on the screen.

dialog_item_prompt *xxx*

or

> *xxx*

Creates a new item for the dialog box and the remainder of the line after "dialog_item_prompt" will be used as the prompt for the item in the dialog box.

dialog_item_var $[0-9] *maxlen*

Specifies which variable $0 through $9 will be used to hold the user's input to this dialog box item. If this entry is omitted, then the dialog box item will be for display of the prompt text only, and the highlight will not be positioned on it for entry by the user. *maxlen* is the maximum number of characters to accept for the value.

Menu Variables

The following variables are maintained with the appropriate values for use with dialog boxes, window titles, item action strings, etc.

$current_window_number

Contains the current window number when the menu was popped up.

$current_window_title

Contains the title of the current window when the menu was popped up.

$version_number

Contains the version number of the Facet*Term* in use.

$ft_hot_key

Expands to a string like "^W" which represents the Facet*Term* window command line hot-key.

$0, $1, $2, ... $9

Variables for which values may be specified by the user via a dialog box. These variables are cleared each time before being entered by the user from a dialog box.

Intrinsic Item Actions

The following are the item actions that are recognized for the **intrinsic** item type.

win_menu

Causes the window selection menu to be built and run with a list of the currently running windows as its list of items. This menu will be a pull down menu without a title.

win_menu_with_title

Same as win_menu except that the menu has a title.

start_cut

Causes Facet*Term* to enter window command mode at the point in the copy operation that the user is marking the corners of the block to be copied.

do_paste

Causes Facet*Term* to paste the contents of the copy buffer into the current window which the menu was popped up over.

print_window

Causes Facet*Term* to print the contents of the current window that the menu was popped up over on the system line printer (or whatever the .facetprint script specifies).

start_shell

Causes Facet*Term* to start the users default shell as specified by the $SHELL environment variable to be started in the next available window.

quit_ft

Causes Facet*Term* to terminate and kill all the processes running in windows.

Window Watch Menu Directives

watch_entries *menu_file*

This special directive is available to create a list of menu items which correspond to all "watch events" that have been loaded from watch files. This list will include all events even if they are currently disabled. Each event is given a letter A-Z to be used as a keyboard short-cut.

You must supply the name of a menu file which will be used to present a list of possible actions to be taken with the selected event.

The following variables are available for use with Window Watch menus:

$alarm_select

This variable expands to the keys necessary to feed to the command line while it is prompting for events to take action on, to get positioned to the event that has been selected from the watch_entries list. See the file /usr/facetterm/text/watchact_menu for examples of use.

Facet*Term* Watch Files (Window Watch)

The Facet*Term* Window Watch feature is controlled by "watch files" which the user loads through the menu, the window command line, or his .facet file. Facet*Term* is installed with a sample watch file which can be made available for system-wide use. The file is named .facetwatch.df and is installed in the /usr/facetterm/text directory. The watch files that are accessed by the menu are installed in the /usr/facetterm/text directory and all file names end with ".fw"

The following is a listing of .facetwatch.df. The listing is divided into topics and each discussed. Any line in a watch file which begins with "#" is considered to be a comment.

The .facetwatch.df File

```
##############################################################################
# example .facetwatch file                           06/18/93
##############################################################################
# ## A FacetTerm "Watch" file specifies conditions that you want
# ##        FacetTerm check periodically and alert you when the
# ##        condition is present.
# ## You load these files into FacetTerm either from the .facet file,
# ##        the FacetTerm command line, or the FacetTerm menu.
##############################################################################

##############################################################################
# ## Line format:
# ##    Lines with "#" in the first column are comments.
# ##    Blank lines are ignored.
# ##    Lines may have leading tabs and/or spaces
# ##    The command (first word on line may be followed by "=" and/or tabs
# ##           and/or spaces.
# ##    Subsequent parameters to the command, if any, may be separated by
# ##           tabs and/or spaces.
```

Overview

```
###############################################################################
# ## Watch entries:
# ##    There are 3 kinds of entries: file, window, and program.
# ##
# ##    Each one of the entries has about a dozen settings that specify
# ##    exactly what it is to watch for, how often, how it is to alarm,
# ##    how often, under what conditions it resets, etc.
# ##
# ##    The example entries below show all of the valid values for each
# ##    of the settings.
# ##    The first value shown for each setting is the default which will
# ##    be used if you do not specify otherwise.
# ##    The default value of optional parameters are shown by a specific
# ##    example with that value for the parameter.
# ##    Settings which do not have a default have the comment:
# ##                   # ## NO_DEFAULT
# ##    just before them.
# ##
# ##    Words that are in all caps indicate a place where you can specify
# ##    whatever value you want, such as:
# ##                   check_interval=SECONDS
# ##    If you want a file to be checked every 3 minutes or 180 seconds,
# ##    put the following in your entry:
# ##                   check_interval=180
# ##
# ##    Words that start with a dollar-sign ($) are variables that you can
# ##    use.  Examples of these are $facetuser which is your username, and
# ##    $file_name which is set to the name you specified on the beginning
# ##    line of a file watch entry.
# ##    Most of these only apply in the specific place that they are shown
# ##    as an example.
```

How it Works

```
###############################################################################
# ## Explanations of settings:
# ##    check_type=
# ##            This tells what in particular to watch for.  For instance,
# ##            if your watch entry is about a file, you might specify that
# ##            you want to watch for the file to be modified.
# ##            Since each of the 3 types of watch entries has its own set
# ##            of check types, they are explained separately below.
# ##    change_status_to_watch=
# ##            This specifies when you want this watch entry to
# ##            automatically go from the "alarm" state back to the
# ##            "watch" state.  For instance, if your watch entry is about
# ##            off screen activity on a window, you may want the alarm
# ##            to automatically go back to watching when you switch to
# ##            that window.
# ##            These are also explained for each type below.
# ##    change_status_to_disable=
# ##            This is the same as change_status_to_watch except it
# ##            automatically sets the state to disabled instead of
# ##            watching.  This is useful when you want to specifically
# ##            enable the watch each time it is of interest and leave
# ##            it disabled otherwise.
# ##            These are also explained for each type below.
# ##    check_interval=
# ##            This specifies how often to check for the condition being
# ##            watched for.
# ##            For files the check_type is looked for every 60 seconds
# ##            by default.
# ##            For programs, the program is run every 300 seconds by default.
# ##            For off screen window activity, the check interval is not
# ##            used since the condition is detected immediately.
# ##    message=
# ##            This specifies what the alarm will say when you see it
# ##            displayed on the command line or the menu.
# ##    alarm_interval=
# ##            This specifies how often the alarm will be displayed
# ##            once the condition you are watching for happens.  This is
# ##            in seconds.
# ##    alarm_type=
# ##            This specifies how you want FacetTerm to inform you that
# ##            the condition that you were watching for has happened.
# ##            The possibilities are:
# ##                    beep
# ##                            This is a distinctive sequence of beeps
# ##                            spaced one second apart.  Default is 3 beeps.
# ##                    command_line
# ##                            This alarm beeps, displays your message
# ##                            between groups of 4 exclamation points,
# ##                            waits a default 3 seconds, beeps and
# ##                            then FacetTerm continues.
# ##                    command_line_quiet
# ##                            Same as command_line without all the
# ##                            beeping.
# ##                    press_return
# ##                            This displays the alarm on the command
# ##                            line and insists that you press return
# ##                            to acknowledge the alarm.
# ##                            This should be reserved for very important
# ##                            events since keys that you press between
# ##                            the occurence of the alarm and your pressing
# ##                            return to acknowledge the alarm are
# ##                            discarded with a beep for each keystroke.
# ##                    paste_to_window=
# ##                            This alarm automatically pastes the specified
# ##                            characters to the specified window.  The
# ##                            window number may be 1 to 10, 0 for window
# ##                            10, L for the last window, or * for the
# ##                            current window.
# ##                            This might be used to trigger a program that
# ##                            is aware of the WindowWatcher facility.
# ##    submenu=
# ##            This is the name of a menu that will pull down when the
# ##            alarm is selected on the watch menu.  It usually gives a
# ##            default action and commands to set or reset the alarm.
# ##            The submenus provided are:
# ##                    mailact_menu $file_name
# ##                            Default action is: mail -f $filename
# ##                    winact_menu $window_number
# ##                            Default action is: Select window $window_number
# ##                    watchact_menu
# ##                            No default action.
# ##    max_alarms_before_quiet=
# ##            This specifies the number of times you want the alarm to
# ##            notify you before is is automatically set to "quiet alarm".
# ##            This is not active unless you specify it.
# ##    max_alarms_before_disable=
# ##            Same as max_alarms_before_quiet except that the alarm is
# ##            disabled after alerting you this many times.
# ##    max_alarms_before_watch=NUMBER
# ##            Same as max_alarms_before_quiet except that the alarm is
# ##            set back to watch after alerting you this many times.
# ##            Note that this only makes sense if the alarm will not
# ##            immediately go off again, such as off screen activity.
# ##    start_enabled
# ##            Watch entries are enabled by default.  Use this directive
# ##            if you want a disabled entry to automatically be enabled
# ##            when this watch entry is loaded over it.
# ##    start_disabled
# ##            Use this directive if you want to create a watch entry
# ##            but you do not want it to start until you specifically
# ##            enable it.
```

Section for Setting up File Watches

```
##############################################################################
# ## To watch for changes in a file, specify "file" followed by the pathname
# ##    of the file.
# ## file_access_not_greater_modified
# ##    A unix file has three times associated with it: the creation time,
# ##    the last modified time, and the last accessed time.  This directive
# ##    is true if the file has been modified since it was last accessed
# ##    or if the modified and accessed times are the same.  This is the
# ##    test used by the C shell "csh".
# ## file_modified_time_change
# ##    This is true if the file has either been modified or has been
# ##    newly created since the last check.
# ##    This is the test used by the Bourne shell "sh".
# ## file_access_greater_modified
# ##    This is true if the file has been read since the last time it was
# ##    modified.
# ## file_zero_length
# ##    This is true if the file size is 0 or the file does not exist.
##############################################################################
# file=FILENAME
# file=/usr/spool/mail/$facetuser

# check_type=file_access_not_greater_modified
# check_type=file_modified_time_change

# change_status_to_watch=file_access_greater_modified
# change_status_to_watch=file_zero_length

# ## NO_DEFAULT
# change_status_to_disable=file_access_greater_modified
# change_status_to_disable=file_zero_length

# check_interval=60
# check_interval=SECONDS

# message=Mail in $file_name
# message=YOUR MESSAGE

# alarm_interval=300
# alarm_interval=SECONDS

# alarm_type=beep
# alarm_type=beep 3
# alarm_type=beep COUNT
# alarm_type=command_line
# alarm_type=command_line 3
# alarm_type=command_line SECONDS
# alarm_type=command_line_quiet
# alarm_type=command_line_quiet 3
# alarm_type=command_line_quiet SECONDS
# alarm_type=press_return
# alarm_type=paste_to_window WINDOW PASTE_CHARACTERS
# alarm_type=paste_to_window L      PASTE_CHARACTERS
# alarm_type=paste_to_window *      PASTE_CHARACTERS
# alarm_type=paste_to_window 1      PASTE_CHARACTERS
# alarm_type=paste_to_window 10     PASTE_CHARACTERS
# alarm_type=paste_to_window 0      PASTE_CHARACTERS

# submenu=watchact_menu
# submenu=mailact_menu $file_name

# ## NO_DEFAULT
# max_alarms_before_quiet=NUMBER

# ## NO_DEFAULT
# max_alarms_before_disable=NUMBER

# ## NO_DEFAULT
# max_alarms_before_watch=NUMBER

# start_enabled
# start_disabled
```

Section for setting up Watches for Offscreen Activity or Pattern Matches

```
##############################################################################
# ## To have FacetTerm watch for offscreen activity, or for a particular
# ##    string to be output on an offscreen window, specify
# ##    "window" followed by the window number.  "0" may be used
# ##    for window 10. "*" may be used to mean the current window.
# ##    Note that "*" may not be used in files loaded from the .facet
# ##    file since there is no current window.
# ## window_offscreen_activity
# ##    This is true when any character is output by the window when it
# ##    is not the current window.
# ## window_offscreen_pattern
# ##    The specified string must be output by the window when it is not
# ##    the current window.  Multiple lines indicating several possible
# ##    patterns may be present.
##############################################################################
# window=1
# window=10
# window=0
# window=*

# check_type=window_offscreen_activity
# check_type=window_offscreen_pattern PATTERN
# ...
# check_type=window_offscreen_pattern PATTERN

# change_status_to_watch=window_is_current_window

# ## NO_DEFAULT
# change_status_to_disable=window_is_current_window

# ## NOT USED - CHECK IS DONE CONTINUOUSLY ## check_interval=0

# message=Window $window_number is active offscreen
# message=THIS IS YOUR ERROR MESSAGE

# alarm_interval=300
# alarm_interval=SECONDS

# alarm_type=beep
# alarm_type=beep 3
# alarm_type=beep COUNT
# alarm_type=command_line
# alarm_type=command_line 3
# alarm_type=command_line SECONDS
# alarm_type=command_line_quiet
# alarm_type=command_line_quiet 3
# alarm_type=command_line_quiet SECONDS
# alarm_type=press_return
# alarm_type=paste_to_window WINDOW PASTE_CHARACTERS
# alarm_type=paste_to_window L      PASTE_CHARACTERS
# alarm_type=paste_to_window *      PASTE_CHARACTERS
# alarm_type=paste_to_window 1      PASTE_CHARACTERS
# alarm_type=paste_to_window 10     PASTE_CHARACTERS
# alarm_type=paste_to_window 0      PASTE_CHARACTERS

# submenu=winact_menu $window_number
# submenu=watchact_menu

# ## NO_DEFAULT
# max_alarms_before_quiet=NUMBER

# ## NO_DEFAULT
# max_alarms_before_disable=NUMBER

# ## NO_DEFAULT
# max_alarms_before_watch=NUMBER

# start_enabled
# start_disabled
```

Section for Setting up Watches Which Run a Program and Check its Return Code

```
##############################################################################
# ## To have FacetTerm periodically run a program to check for alarm
# ##    conditions, specify "program" followed by the pathname of
# ##    the program.
# ## program_exit_status_0
# ##    When the program specified was run, it executed properly and
# ##    exited with the return code 0 meaning true.
# ## program_exit_status_non_0
# ##    The specified program could not be executed or exited with
# ##    a non zero return code meaning false.
##############################################################################
# program=PATHNAME

# check_type=program_exit_status_0

# change_status_to_watch=program_exit_status_non_0

# ## NO_DEFAULT
# change_status_to_disable=program_exit_status_non_0

# check_interval=300
# check_interval=SECONDS

# message=Mail from $program_name
# message=This is an alarm message

# alarm_interval=300
# alarm_interval=SECONDS

# alarm_type=beep
# alarm_type=beep 3
# alarm_type=beep COUNT
# alarm_type=command_line
# alarm_type=command_line 3
# alarm_type=command_line SECONDS
# alarm_type=command_line_quiet
# alarm_type=command_line_quiet 3
# alarm_type=command_line_quiet SECONDS
# alarm_type=press_return
# alarm_type=paste_to_window WINDOW PASTE_CHARACTERS
# alarm_type=paste_to_window L      PASTE_CHARACTERS
# alarm_type=paste_to_window *      PASTE_CHARACTERS
# alarm_type=paste_to_window 1      PASTE_CHARACTERS
# alarm_type=paste_to_window 10     PASTE_CHARACTERS
# alarm_type=paste_to_window 0      PASTE_CHARACTERS

# submenu=watchact_menu
# submenu=mailact_menu $program_name
# ## NO_DEFAULT
# max_alarms_before_quiet=NUMBER

# ## NO_DEFAULT
# max_alarms_before_disable=NUMBER

# ## NO_DEFAULT
# max_alarms_before_watch=NUMBER

# start_enabled
# start_disabled
```

File Inclusion Feature to Allow Modular Development of Watch Files

```
##############################################################################
# ## To include another file at this position, specify "use" followed by
# ##         the filename.
# ##         The filename may optionally be followed by parameters to be
# ##         used in the watch_file.
# ##         Each watch entry must be self-contained and may not span files.
##############################################################################

# use=WATCH_FILENAME
# use=WATCH_FILENAME PARAMETERS

##############################################################################
```

Changing Facet*Term* Messages

Most of the text strings output to the terminal by Facet*Term*, such as prompts or user errors, can be redefined. For example, you may want to use this feature to change the language to something other than English. This can be accomplished for the facetterm program by modifying a file named facettext. Messages from the Facet*Term* menu (other than the contents of the menus themselves) can be changed by modifying the facetuitext file.

If the facettext or facetuitext files exist, Facet*Term* will read these files looking for lines of the form:

!*text_string_name*

where the "!" must be in column 1, and the *text_string_name* matches one of the names of the strings that can be redefined.

Upon finding such a line, the contents of the next line that does not begin with a "#" replaces the default contents of the text string.

For example, a facettext file that contains the following:

```
!facet_activating
#Facet process: activating\r\n
FacetTerm is now activating sessions...\r\n

!facetterm_window_label
#FacetTerm Window %d\n
FacetTerm Session %d \n
```

will change the default messages shown on the lines starting with the "#" to the new contents shown on the line after them.

The file /usr/facetterm/text/facettext.ex contains all of the valid text string names and the default contents of the strings, each with a "#" in front of them to make them comments.

The recommended method for redefining the text strings is:

1. Copy /usr/facetterm/text/facettext.ex to the desired directory (see the earlier section on the Facet*Term* text file

search path), and remove the ".ex" suffix.

2. Edit the facettext file using a text editor, such as vi.

3. Duplicate the line containing the default contents of the message which you want to change.

4. Remove the "#" in column 1 of the new line, and make the desired changes to its contents.

5. Remove the "#" in column 1 of the line containing the "!" and the text_string_name.

The processor for these lines understands:

\r	for carriage return
\n	for line feed
\s	for space
\\	for backslash

Since the terminal is in raw mode when most of these strings are output, you will see the combination "\r\n" used for going to the beginning of the next line. "\s" is used for trailing spaces to make them visible in the file.

As you can see by the default contents of the text strings, most of them are input to printf() or sprintf() with parameters to fill in window numbers, etc. Changing the "%" fields of the strings should be done only by programmers aware of the consequences of making an invalid call to sprintf().

Making Facet*Term* Function Key Files

Function key files tell Facet*Term* how to use a terminal's function keys. Each line of the function key file contains:

1. The function key number (starting at 1 for the first programmable key. This is usually F1, but may be something else such as F6 on a VT220).

2. A single space or single tab.

3. The string the function key should be programmed to send (see below for syntax of control and special characters).

Blank lines and lines beginning with "#" are treated as comments and are ignored.

For example:

```
# program the first function key to send
# 'h' 'e' 'l' 'l' 'o' 'carriage-return'

1  hello\r

# program the third function key to send
# 'escape' 'a' 'b' 'c'

3  \Eabc
```

The function key number can be entered as "S1" for shifted function key number 1, if the terminal description file for the terminal indicates that there are shifted function keys. If it does not, the "S" is ignored. For example:

```
# program the fifth shifted function key to send
# 'Control-C' 'x'

S5  ^Cx
```

If your terminal has function key labels, the function key number can be entered as "L1" for the label on function key 1. For example:

```
# program the third function key to have the
# label "list" and to send "ls -l"

L3  list
3  ls -l
```

Facet*Term* looks for the function key file in the following directories (the same path by which terminal description files are found):

1. The directory in which Facet*Term* was started.

2. The $FACETINFO directory.

3. The $HOME directory.

4. The /usr/facetterm/localterm directory.

5. The /usr/facetterm/term directory.

6. The /usr/facetterm directory.

The following table gives the syntax for special characters in a function key string.

NAME	HEX	SYNTAX
Backslash	5C	\\
octal value	00	\000
.	.	.
.	.	.
Escape	1B	\E
Return	0D	\r
Newline	0A	\n
Backspace	08	\b
Caret(^)	5E	\136
Delete	7F	^?
Control-@	00	^@
.	.	.
.	.	.
Control-Z	1A	^Z
Space	20	\040

Utility Programs for Use With Facet*Term*

Several utility programs are included with Facet*Term* which you may find useful in shell scripts. The usage of each of these is described below.

Sending Window Commands to Facet*Term* Programmatically With fct_command

The fct_command program allows you to issue window commands to Facet*Term* from a shell script or the shell command line. When running this command, stdin must be a Facet*Term* window. Its usage is:

fct_command '*command 1***' ... '***command n***'**

where '*command 1*' through '*command n*' are strings which are valid commands to the Facet*Term* window command line. The quotes are only needed around the commands if there are any characters in the command which would be interpreted by the shell, such as an embedded blank or asterisk. For example,

```
fct_command t1 b2
```

will put window 1 in the top half of a split screen and window 2 in the bottom half of a split screen.

Special characters are indicated in the command strings in the following manner:

Return	\r
Escape	\E
Backslash	\\
0x00	^@
...	
0x1F	^_

The return code from fct_command is 0 if all of the window commands were successfully sent to Facet*Term* Otherwise, it prints a diagnostic and returns non-zero.

Getting Information About the Facet*Term* Environment With fct_info

The fct_info command allows you to get information about the Facet*Term* environment. Its usage is:

fct_info *infoname*

where infoname is one of the following:

not_a_window

Causes fct_info to return 0 if stdin is not a window and 1 if is a window.

is_a_window

Causes fct_info to return 1 if stdin is not a window and 0 if is a window.

ttyname

Causes fct_info to echo to stdout the name of the real tty that Facet*Term* is running on. If stdin of fct_info is not a window, it prints a diagnostic and returns non-zero.

window_number

Causes fct_info to echo to stdout the number of the window that it is running on. If stdin of fct_info is not a window, it prints a diagnostic and returns non-zero.

number_of_windows

Causes fct_info to echo to stdout the maximum number of windows that Facet*Term* will use. If Facet*Term* was started with no limit to the number of windows, then this number will be 10. Otherwise, it will be the number of windows that Facet*Term* was limited to when it was started. If stdin of fct_info is not a window, it prints a diagnostic and returns non-zero.

ptyname_*n*

Causes fct_info to echo to stdout the name of the device that is being used for window *n* where *n* is 1 through 10 (do not use 0 for window 10). If stdin of fct_info is not a window, it prints a diagnostic and returns non-zero.

facet_type

Causes fct_info to echo to stdout FACET/TERM if running on a Facet*Term* window, and FACET/PC if running on a FacetPC window. If stdin of fct_info is not a window, it prints a diagnostic and returns non-zero.

current_window_number

Causes fct_info to echo to stdout the current window number (not necessarily the window it is running on). If stdin of fct_info is not a window, it prints a diagnostic and returns non-zero.

active_window_numbers

Causes fct_info to echo to stdout a list of active windows separated by spaces. If stdin of fct_info is not a window, it prints a diagnostic and returns non-zero.

idle_window_numbers

Causes fct_info to echo to stdout a list of idle windows separated by spaces. If stdin of fct_info is not a window, it prints a diagnostic and returns non-zero.

program_name

Causes fct_info to echo to stdout the title of the window that it is running on. If stdin of fct_info is not a window, it prints a diagnostic and returns non-zero.

program_name_*n*

Causes fct_info to echo to stdout the title of window *n* where *n* is 1 through 10 (do not use 0 for window 10). If stdin of fct_info is not a window, it prints a diagnostic and returns non-zero.

hotkey

Causes fct_info to echo to stdout the window command line hotkey in a printable format (such as ^W for control-W). If stdin of fct_info is not a window, it prints a diagnostic and returns non-zero.

If the *infoname* supplied to fct_info is not one of the above, it simply echoes a newline to stdout.

Facet*Term* Window Command Line Reference

This chapter is a complete reference to all of the commands accepted by the Facet*Term* window command line. All of the tables in this chapter assume that the window command line has been entered by pressing the hot-key. The default hot-key is:

CTRL W

All commands terminate window command line mode unless otherwise noted.

Window Selection

1 2 ... 9 0	Select window 1 through window 10
- or BACKSPACE or ←	Select the previous active window (remains in window command line)
+ or = or →	Select the next active window (remains in window command line)
L	Select the last window that you were on

Miscellaneous Common Commands

RETURN	Refresh the current window
□ or ESC	Exit the window command line without performing any action
O	Output the screen from the current window to the printer using the .facetprint script (Screen print)
? or H	View help screens
CTRL W	Pass ^W (hot-key) on through to the current window
Q Y	Quit Facet*Term* after terminating any programs running in active windows.

Starting Programs

S	Start a shell on an idle window. If the current window is idle, it will start there. Otherwise, the first available idle window will be chosen.
R *program* RETURN	Start a program on an idle window. The program name and any start-up parameters should be specified as if starting the program from a shell.
A	Re-activate the current window. If the current window has a start-up program specified in the .facet file, then that program is started to re-activate the window. Otherwise, the last program run in the window is started. If the current window is not idle, the command is ignored.

Copy and Paste

Copy Operations

The following commands perform a copy operation and choose the type of copy.

C *position 1* RETURN *position 2* RETURN	Copy characters from the current window. See below for position keys.
	While in copy mode, you may press the following keys to select the type of copy:
B	To select a block copy
V	To select a vertical copy
S	To select a stream copy
W	To select a wrap copy

Positioning While in a Copy Operation

The following keys may be pressed while in a copy operation to move the cursor for the purpose of marking the area to be copied.

↑ or U	Move up one character
SHIFT U	Move up to the top row
↓ or D	Move down one character
SHIFT D	Move down to the bottom row
← or L	Move left one character
SHIFT L	Move left to the first column
→ or R	Move right one character
SHIFT R	Move right to the last column
HOME	Move to the first column and first row

Marking Corners While in a Copy Operation

The following keys may be pressed while in a copy operation to mark corners of the area to be copied.

RETURN	Mark a corner at the current position
M	Re-mark the first corner at the current position
C	Change which corner is being marked (switch from corner 1 to corner 2 or from corner 2 to corner 1)
O	Show the outline of the copy area

Commands to Mark Entire Sections

The following keys may be pressed while in a copy operation to mark the entire section indicated.

A	Marks all characters on the window from the upper left corner to the lower right corner.
E	Marks all characters on a line from the current cursor position to the end of the line.

In both cases, you will still be left in the copy operation and must press

RETURN

to finish the copy.

Paste Commands

The following commands paste the contents of the copy buffer to the indicated destination.

P	Paste the contents of the copy buffer to the current window.
> P	Paste the copy buffer to the printer using the .facetprint script.
> S *script* RETURN	Paste to a script named *script*.
> F *file* RETURN	Paste to a file (cannot already exist).
> A *file* RETURN	Paste to a file (append or create).
> O *file* RETURN	Paste to a file (overwrite or create).
E *type* RETURN	Allows specification of an alternate type of end of line character during paste. For example, if you are pasting into a spreadsheet which requires a down arrow to terminate entry of a cell, you would need to paste a down arrow at the end of each line rather than a Return character. Full implementation of this feature requires modification of terminal description files to suit the particular needs of your application. If you need this feature, call a Facet*Corp* support engineer for assistance.

"Window Watch" Commands

W F *file* RETURN	Load a watch file.
W Q	Quiet all alarms which are currently going off.
W W	This command will cause Facet*Term* to ask, for each event being alarmed, if you want to reset to watch for the next event.
	The following commands will cause Facet*Term* to ask, for each event being watched, whether you want the specified action taken on that event.
W D	Disable a watch.
W E	Enable a watch.
	When FacetTerm is asking for verification to take action on a series of events from the W D or E commands, besides giving the answer being prompted for, you can press:
Q	to quiet an event with its alarm going off
W	to start watching for the event
E	to enable the event
D	to disable the event

Programming Function Keys

G *file* RETURN	Load a function key file globally for all windows.
K *file* RETURN	Load a function key file for the current window only.
	file may be the name of your own function key file, or one of the following names which have special meaning:
DEFAULT	means to load the default values for the terminal.
WINDOWS	means to load a set of values useful for selecting windows.
+	means reload the keys to the values that were set by "function_keys=" in the .facet file and by load function key commands.
-	means reload the keys to the DEFAULT without changing the values that will be reloaded by **+**.

Special Keys

. S	Sends a ^S to the current window. If your terminal is running in Xon-Xoff mode, pressing an actual ^S will stop the entire line, not just the current window.
. Q	Sends a ^Q to the current window.
. @	Sends a Null (^@) to the current window.
. B	Sends a BREAK condition to the current window (not supported on some systems).

Key Mapping

: H *key* *map* RETURN	This command allows you to create a "hot-key" such that when the specified *key* is pressed while on any window, the string specified by *map* is sent to the window which is current when this command is done.
: M *key* *map* RETURN	This command provides a simple macro facility, allowing you to specify that when *key* is pressed, the string specified by *map* is substituted. This key mapping will occur on all windows.
: U *key* *map* RETURN	This command allows you to unmap a key that has been set-up as a hot-key or macro key with either of the previous commands.
	In the case of all these commands, control keys should be specified by using the "^" character. You cannot actually type a control key to the window command line. When a *key* and *map* are both specified, there should be no space between the two.

Split Screen

T	Change the next selected window to be in the top half of the screen.
B	Change the next selected window to be in the bottom half of the screen.
F	Change the next selected window to be full screen.
↑ or U	Move the split screen divider up one line.
↓ or D	Move the split screen divider down one line.

All of the split screen commands leave you in window command mode.

Window Titles

' *title* RETURN or " *title* RETURN	Change the title of the current window to *title*.

Terminal Modes

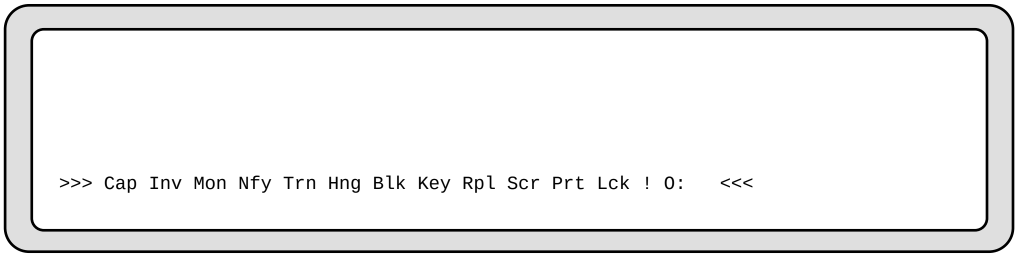
M *settings* RETURN	Changes terminal mode settings. Modes are terminal dependent, and some terminal descriptions do not include modes. *settings* are made with the following keys. You may make as many setting changes as you want before terminating the mode changes with Return.
↓ or D	Select next mode name.
↑ or U	Select previous mode name.
→ , N or R	Select next setting for the mode currently being displayed.
← , P or L	Select previous setting for the mode currently being displayed.

"Extra" Commands

The window command line has an "extra" layer of commands which may be accessed by pressing:

X

while in the window command line.   The command line will change to show:



```
>>> Cap Inv Mon Nfy Trn Hng Blk Key Rpl Scr Prt Lck ! O:   <<<
```

The tables in this section contain all of the extra commands.

Capturing Window Output

X C Y *file* RETURN	Start capturing the output of the current window to *file*.cap
X C N *file* RETURN	Turn off capture for the current window.

Making a Window Invisible to a Selection Scan

X I *window#* Y	Make window *window#* invisible to the active window scan commands.
X I *window#* N	Restore window *window#* to be visible to the active window scan commands.
	In the case of both commands, *window#* may be omitted, and the current window will be acted on.

Monitor Mode

Monitor mode causes all characters to be displayed on the screen in a printable format, and no escape sequence commands are processed. This is primarily a support feature.

X M *window#* Y	Put window *window#* into monitor mode.
X M *window#* N	Take window *window#* out of monitor mode.
	In the case of both commands, *window#* may be omitted, and the current window will be acted on.

Notify When Current

This feature is a programmer's feature and is used by the Facet*Term* menu. Turning this feature on for a window causes Facet*Term* to send a "^Y" character to the standard input of the window when the window is selected as the current window.

X N *window#* Y	Have Facet*Term* notify window *window#* when it is selected as the current window.
X N *window#* N	No longer have Facet*Term* notify window *window#* when it is current.
	In the case of both commands, *window#* may be omitted, and the current window will be acted on.

Transparent Mode

Transparent mode causes all characters output to a window to be passed on through to the terminal with no recognition by Facet*Term*. This is primarily a support feature.

X T *window#* Y	Put window *window#* into transparent mode.
X T *window#* N	Take window *window#* out of transparent mode.
	In the case of both commands, *window#* may be omitted, and the current window will be acted on.

Sending Signals

You can send signals to programs running in Facet*Term* windows with the following commands. The effect of these signals is usually to terminate the application running in the signaled window.

X H Y	Sends a hang-up signal (SIGHUP) to the current window.
X H A	Sends a hang-up signal (SIGHUP) to all windows.
X H K	Sends a kill signal (SIGKILL) to the current window.

Block Window When Not Current

In its normal mode of operation, Facet*Term* continues to process the output from a window even when the window is not the current window. This options allows you to block the output from the window when the window is not the current window. This will usually cause the process to be suspended after some time waiting for its output to drain.

X B *window#* Y	Block output on window *window#* when the window is not current.
X B *window#* N	Take window *window#* out of blocked mode.
	In the case of both commands, *window#* may be omitted, and the current window will be acted on.

Keystroke Capture and Replay

Facet*Term* can capture keystrokes and save them to a file for replay later. Unlike the capture of output which is only for a particular window, keystrokes to any window plus the window command line or menu will be captured. This is primarily a support feature.

X K Y *file* RETURN	Start capturing keystrokes and save them in a file named *file*.key.
X K N	Stop capturing keystrokes.
X R *file* RETURN	Replay the keystroke capture file named *file*.key.

Screen Saver and Screen Lock

Facet*Term* has a screen saver feature for use on those terminals (or PC consoles) which do not have a built-in screen saver. When the screen saver is enabled, it will time periods of inactivity on the keyboard. When the screen saver period has expired, the screen is blanked, and a small message is moved around on the screen periodically. The first key typed on the keyboard when the screen saver is on will be ignored, and the screen saver will be turned off, the screen refreshed with the current window, and the timer re-started. The message and the time-out period can be set in the .facet file.

Screen lock is similar except that it is not automatically turned on by a timer, and it requires entry of a password to turn it on and off.

X S RETURN	Puts Facet*Term* into screen saver mode if the screen saver is enabled.
X S Y	Enables the timed screen saver and puts Facet*Term* into screen saver mode.
X S N	Disables the timed screen saver.
X L Y *lock* RETURN *lock* RETURN	Turn on the screen lock. Notice that you must type the same *lock* word twice. You must type the same *lock* word again to turn off the screen lock.

Printer Mode

If your terminal supports transparent print to an attached printer, you may use the window command line to put a window in and out of transparent print mode. When in transparent print mode, all output from a window is directed to the attached printer. If your application switches the terminal in and out of transparent print itself, it is not necessary to use the window command line to switch the mode.

X P *window#* Y	Put window *window#* into transparent print mode.
X P *window#* N	Take window *window#* out of transparent print mode.
	In the case of both commands, *window#* may be omitted, and the current window will be acted on.

Screen Print Using a User Specified Script

This command is exactly like ^W O, in that it is used to output the current window, except that it allows you to specify the script that will be run instead of using the .facetprint script.

X O *script* RETURN	Place the contents of the current window in temporary files and Run the script named script, giving it the name of the temporary files as arguments $1 and $2.

As with the standard screen print feature, the first temporary file passed will simply contain the characters from the screen, while the second file will contain the information necessary for the fct_printx program to print special characters.

Suspend Facet*Term* and Start a Program on the Line

This command allows you to suspend your entire Facet*Term* session, and start a program on the line. When you exit the program, Facet*Term* will resume and refresh your screen with the current window. This feature is useful when you are using a terminal emulator on a PC, and want to do a file transfer. You cannot do the file transfer through a Facet*Term* window, because special characters which might be part of a binary file being transferred or part of the file transfer protocol would be interpreted by Facet*Term* rather than being passed through. However, rather than having to shutdown your entire Facet*Term* session, you can suspend it and then resume it after the file transfer is done.

X ! *program* RETURN	Suspend Facet*Term* and run *program* on your terminal's actual tty device. When you exit the program, Facet*Term* will resume.

Setting the Windows Window

The "windows window", if set, is selected when a break key is pressed. If the windows window is not set, the window command line comes up when a break is pressed.

X W *window#* RETURN	Set the windows window to *window#*.
X W N RETURN	Turn off the windows window feature.

Window Control Commands

These commands may be used to automatically control the selection of windows and coordinate with the Facet*Term* environment. They are primarily for use by programs such as the Facet*Term* menu, and are not normally used from the keyboard.

Setting the Idle Window

If set, the idle window is the window which is selected when any window goes idle. Alternatively, the idle window feature may be set to send the idle window a ^Z character to notify it that a window has gone idle.

I *window#* RETURN	Set the idle window to *window#*, thus causing this window to be selected when any window goes idle.
I A RETURN	Causes Facet*Term* to select the next active window when a window goes idle.
I *window#* M RETURN	Causes Facet*Term* to send a ^Z to *window#* when a window goes idle.
I N RETURN	Turn off the idle window feature.

Index