3.3 Release Notes: The Terminal Application

This file contains release notes for the 3.3, 3.1, and 3.0 releases of Terminal. Items specific to the 3.3 release are listed first, and other notes follow.

Notes Specific to Release 3.3

New Features

The following new features have been added to Terminal since Release 3.1:

- Execution service

  Terminal now publishes a Distributed Objects interface to allow it to execute commands on the behalf of other applications. See **/NextDeveloper/Headers/apps/Terminal.h** for more information.

- Color setting

You can drag and drop colors into Terminal windows to allow reverse video and other interesting settings. The color of the selection is settable, as are the color and size of the cursor.

- Output indication

  When Terminal is hidden or windows are miniturized, the text in their tiles changes and scrolls to indicate that output has been received that has not been viewed.

- Settable title bar text

  A process can set the title bar of its terminal window by emitting the following string, without spaces or quotes:

  ESC] 0 ; "Title String" BEL

  where ESC is '\033', and BEL is '\007'. The initial 0 may alternatively be a 2.

- Meta key handling

  When the a key is pressed in conjunction with the Alternate key, there is yet another option for Meta key handling. With the following dwrite:

  dwrite Terminal Meta 0

  The Meta key generates a character with the eighth bit set. The other options for Meta (27 and -1) generate the key preceded by an escape character, or standard NeXTSTEP deadkey handling, respectively.

- Keyboard accelerators

  A few keyboard accelerators have been added so power users won't have to take their hands off the keyboard quite as often:

  Alt-(up & down arrows) - scroll up or down a line
  Alt-Shift-(up & down arrows) - scroll up or down a page
  Alt-(left & right arrows) - switch to next (previous) terminal window
  Alt-Shift-(left & right arrows) - same as above, but can unminiturize windows

- VT100 smooth scrolling

  The VT100 smooth scrolling mode (available via Escape codes) has been added, though the window reverts to jump scrolling if display falls behind.

- Environment flags

  It is sometimes useful to know the tool that is providing the terminal emulation without changing the TERM environment variable. Terminal now sets two environment variables to provide this information to subprocesses:

  TERM_PROGRAM=NeXT_Terminal
  TERM_PROGRAM_VERSION=64.1 (or something else...)

- A tool for creating new shell windows

  The program **/usr/bin/terminal** may be useful for opening new shell windows in the current (or another) directory. Without any arguments, it will open a new shell window in the current directory. (However, you may have a .login or .cshrc file which defeats this by changing to the user's home directory; you may wish to change this.) If a command is specified, terminal

will, by default, run the command without a shell for fastest startup. If a shell is desired, it can be guaranteed by running the command with the -shell option. For more information, type "/usr/bin/terminal -help".

Bugs Fixed in Release 3.3

These bugs have been fixed:

- Selection no longer disappears when incoming text appears
- No more size restriction on search string
- Background Terminal services no longer activate Terminal
- VT100 bold text can really be bold rather than dim
- VT100 emulation and attribute handling has been made slightly more faithful
- "find text" positions text in window in a more reasonable manner
- A rare race condition crasher was fixed

3.3 Terminal Feature examples

The following **csh** script code (which can be added to a **.cshrc** file) shows one example of how you can set the title bar to a "clipped" version of the current directory. It also puts the hostname in the prompt in inverse video:

```
# begin csh script to set the title bar to the current working directory
set homedirectory = ~${user}
alias shortpwd 'pwd|sed -e "s|^/private||" -e "s|${homedirectory}|~${user}|"'
alias getTitle 'set titleString="`shortpwd|sed -e '\''s|.*[^/]\(/[^/]*/[^/]*/[^/]*/[^/]*\)|...\1|'\'' `"'
getTitle

if( ${?prompt} ) then
    set hostNm=`hostname`
```

```
    alias cd 'cd \!*; getTitle ; echo -n "]0;${titleString}" ; set prompt = "[7m${hostNm} >[m "'
endif
cd .
#end csh script
```

3.3 Terminal Execution Service example

The following header should be identical to **/NextDeveloper/Headers/apps/Terminal.h**, but is provided in case that file has not been made available. If they differ, the file in **/NextDeveloper/Headers** supercedes the contents below:

Terminal.h ¬

The following file, **terminal.m**, demonstrates how the Terminal protocol can be used to create a command line utility to startup shell windows:

terminal.m ¬

Notes Specific to Release 3.1

New Features

The following new features have been added to Terminal since Release 3.0:

- Window position saving

  To facilitate the use of Terminal on computers with small screens, the location of Terminal

windows is now saved relative to the dimensions of the screen. On a larger or smaller screen, the windows should appear in approximately the same relationship.

The format of Terminal configuration files has been modified to support this change. Configurations saved under Release 3.1 can't be used on a Release 3.0 system.

- VT100 function key emulation

The location of VT100 function keys has changed. When VT100 keypad emulation was enabled in Release 3.0, the top row of keys on the numeric keypad could be used to generate the codes PF1 through PF4. In Release 3.1, the numeric keys '1' through '4' assume this function. Keys '4' through '9' continue to generate numbers.

When Terminal is used with an application that employs the VT100 "application keypad" mode, the number keys on the numeric keypad revert to generating codes appropriate to this mode. To get PF1 through PF4, use Shift-1 through Shift-4.

As before, the Alternate key may be used to toggle the state of VT100 keypad emulation as a particular character is typed. Thus, Alternate-1 generates the character '1' when VT100 keypad emulation is on, and it generates the code for PF1 when VT100 keypad emulation is off.

Notes Specific to Release 3.0

These notes were included with the Release 3.0 version of Terminal. Sections that are no longer relevant have been marked with an italicized comment.

New Features

The following new features have been added to Terminal since Release 2.0:

· Performance improvements

There have been a number of small improvements, along with an overhaul of Terminal's memory management system. Creating, scrolling, and closing windows are the operations most affected.

· Preferences panel

There are a few new options, and settings on existing windows can now be changed. The Preferences panel has been subdivided into different subpanes, some of which apply to individual windows (or defaults) and some of which apply globally.

The new preferences are: control over the maximum size of the scrollback buffer (you now don't need a scroller well if you have no buffer), the ability to make windows "sticky" so they don't disappear when the shell inside exits, customizable title bars, process monitoring (see below), and some autolaunching options.

· Saved windows

A window or set of windows can be saved to a file, allowing you to save your preferred configurations. Everything about the windows is saved except the contents of the scrollback bufferÐthis includes the shell, the location of the window, and whether or not the window is miniaturized. When you Save As, you choose whether you would like all windows or just the main window saved.

Once a window is associated with a file, Save can be used to flush the settings out without seeing a Save Panel, just as with more conventional documents. However, if more than one window belongs with that file, *all* the relevant windows will be re-saved (the Save menu item indicates this by changing its label to Save Set). This convention allows you to open a set of windows, rearrange them, and type Command-s to boink them back to the file.

To have a file opened for you automatically each time you start up Terminal, you can either check off a box to this effect when you Save As, or you can specify the filename in the Preferences panel under Startup options.

· Activity monitoring

Terminal tries to determine whether your shell windows are in active use by monitoring the processes inside them. If Terminal thinks something interesting is going on inside a window, it will mark the window with a broken X. You'll be prompted for confirmation before closing a "busy" window or quitting Terminal when there are busy windows.

To make this evaluation, Terminal collects **ps**-type information about processes it considers relevant and applies a number of heuristics. Usually, Terminal considers shells and a few other garden-variety processes such as **su** to be innocuous and will not mark windows busy on their account. But there are exceptions; for example, if the distinguished process on the tty is a shell but it is actively running, the window will be dirty. You may designate additional "clean" command names (**rlogin** and **telnet** are common) in the preferences panel. If there are running processes on the terminal, the terminal might still be clean if the processes seem to be running happily in the background. But an interesting process that's the current process or that you've explicitly suspended with Control-z will always make the window dirty.

The status of windows is checked only when it is likely to have changed. There is a back-off strategy which prevents the probes from happening too often if the window seems constantly busy.

· New accelerators

Two panels, Quick Title and New Command, provide an easy way to set the title on a window and to start up a new window running a designated command. Both of these would otherwise require use of the Preferences panel.

Quick Title provides the current title of the window for you to edit; however, when you set the title this way Terminal won't update any of the information in the future. For example, if your window is titled "/bin/csh 80x24 ~/Term.term" and you edit that to include some text of your own choosing, the new title is treated as a string and won't reflect (for example) the size of the window should you change it. The preferences panel allows better control over this; you can mix your own material with Terminal's automatically updated information.

New Command does use path searching when it execs your command, but since the path is just the default inherited from Workspace, it's utility is limited. You can add arguments, but shell metacharacters won't be handled correctly.

· Terminal services

These are like Pipes and Commands in Edit, only they work for all applications and there is a user interface for constructing them. Terminal services are dynamic and appear in other apps' Services menu as soon as you define them.

When specifying the command associated with a service, you can use the tokens %s and %p to refer to the location where the selection and prompted input are inserted, repsectively (prompted input is not requested unless this string appears in the command).

· Dragged icons

Terminal can accept files dragged into a window. The appropriate filenames will be pasted, plus a space.

- Miscellaneous changes and new features

Some menu items have been moved around. There is no longer a Format menu. Copy Font and Paste Font work now. The print panel has been improved. Shells that die or can't be executed are handled much better. app:openFile.... can be used to create new windows running a given command or shell (Terminal replaces the Workspace shell in 3.0).

Japanese Version

Users of Release 2.1J will notice the following changes:

- Front end processor input is blocked for non-Japanese fonts
- Very minor variations in front end processor intermediate display
- Selection- and cursor-related bug fixes
- Performance improvements
- A warning is given for font changes which would garble Japanese data

When a window's font is changed from Japanese to non-Japanese there is the potential that existing Japanese characters will not be displayed correctly. A warning appears in this case, giving the user the choice of cancelling or proceeding with the font change. However, if all data in the window is ASCII text, then no warning appears. Changing back to a Japanese font will display the garbled text correctly.

About Japanese EUC support

The Japanese version of Terminal does not support the extended non-ASCII characters of Next Standard Encoding, which are not generally used in Terminal. Any non-EUC data received is converted to '?'.

The Japanese version of Terminal processes EUC only, regardless of font selection. Use of a Latin font such as Courier works for ASCII text, but any non-ASCII characters are still processed as EUC codes: line breaking and illegal data detection will still be as for EUC.

The VT100 Emulation Alternate-key options in Terminal Preferences should always specify escape sequences when using Japanese, as the other option generates codes that conflict with EUC. Escape sequences are the default in the Japanese version of Terminal.