

NAME

`mintty` - Cygwin terminal emulator

SYNOPSIS

`mintty` [*OPTION*]... [- | *PROGRAM* [*ARG*]...]

DESCRIPTION

Mintty is a terminal emulator for Cygwin with a native Windows user interface and minimalist design. Its terminal emulation is largely compatible with **xterm**, but it does not require an X server.

INVOCATION

If a program name is supplied on the command line, this is executed with any additional arguments given. Otherwise, `mintty` looks for a shell to execute in the *SHELL* environment variable. If that is not set, it reads the user's default shell setting from */etc/passwd*. As a last resort, it falls back to */bin/sh*. If a single dash is specified instead of a program name, the shell is invoked as a login shell.

OPTIONS

The standard GNU option formats are accepted, with single dashes introducing short options and double dashes introducing long options, e.g. **-o arg**, **--option arg**, and **--option=arg**.

-e, --exec PROGRAM [ARG ...]

Execute the specified program in the terminal session and pass on any additional arguments. This option can be omitted, in which case the first non-option argument, if any, is taken as the name of the program to execute.

-c, --config=FILENAME

Use the specified configuration file instead of the default *~/minttyrc*.

-p, --position=X,Y

Open the window with its top left corner at the specified coordinates.

-s, --size=COLS,ROWS

Set the initial numbers of character columns and rows.

-t, --title=TITLE

Use *TITLE* as the initial window title. By default, the title is set to the executed command.

-i, --icon=FILE

Load the window icon from a Windows *.ICO* file.

-l, --log=FILE

Copy all output into the specified log file. (See *script(1)* for a more flexible logging tool.)

-u, --utmp

Create a utmp entry.

-h, --hold=never|always|error

Determine whether to keep the terminal window open after the command exits. The argument can be abbreviated to a single letter. By default, the window is closed immediately. Alternatively, it can be set to always stay open, or to stay open only if the command finished with a non-zero exit code or due to a runtime error signal.

-H, --help

Display a brief help message and exit.

-V, --version

Print version information and exit.

USAGE

Mintty aims to adhere to both Windows and Unix usage conventions. Where they conflict, the Windows conventions generally take precedence, at least in the default configuration. This is what this section primarily describes; see the **CONFIGURATION** section on how it can be customised.

Menus

As usual, the context menu can be opened by right-clicking the mouse or by pressing the **Menu** key that is normally located next to the right Ctrl key.

Mintty also adds a couple of items to the window menu, which can be accessed by clicking on the program icon or pressing **Alt+Space**.

Both menus have an entry that leads to the options dialog for changing mintty's configuration.

Copy & paste

Screen contents can be selected by holding down the left mouse button and dragging the mouse. The selection can be extended by holding down **Shift** or **Ctrl** while left-clicking. Double-clicking or triple-clicking selects a whole word or line, whereby word selection includes special characters that commonly appear in file names and URLs.

In the default configuration, selected text has to be explicitly copied to the clipboard using either the **Copy** menu command, the **Ctrl+Ins** keyboard shortcut, or the middle mouse button combined with **Shift**. X-like copy-on-select behaviour can be enabled on the **Mouse** page of the options dialog.

The selected region is copied as "rich text" as well as normal text, which means it can be pasted with colours and formatting into applications that support it, e.g. word processors.

The clipboard contents can be pasted using either the **Paste** menu command, the **Shift+Ins** keyboard shortcut, or the middle mouse button. Not only text but also files and directories can be pasted, whereby the latter are inserted as quoted filenames.

Drag & drop

Text, files and directories can be dropped into the mintty window. They are inserted in the same way as if they were pasted from the clipboard.

Opening files, directories and URLs

Files, directories and URLs can be opened either by holding **Ctrl** while left-clicking on them, or by selecting them and choosing the **Open** command from the context menu. Please note that opening a file or directory with a relative paths only works correctly if the path refers to the current working directory of the process invoked by mintty.

Font zoom

The font size can be increased or decreased using the keyboard shortcuts **Ctrl+plus** and **Ctrl+minus**. **Ctrl+zero** returns it to the size configured in the options dialog.

Fullscreen

Fullscreen mode can be toggled using either the **Fullscreen** command in the menu or the **Alt+Enter** and **Alt+F11** keyboard shortcuts.

Default size

If the window has been resized, it can be returned to the default size set in the Window pane of the options using the **Default size** command in the menu or the **Alt+F10** shortcut.

Reset

Sometimes a faulty application or printing a binary file will leave the terminal in an unusable state. In that case, resetting the terminal via the **Reset** command in the menu or the **Alt+F8** keyboard shortcut should help.

Scrolling

Mintty has a scrollbar buffer that can hold up to 10000 lines in the default configuration. It can be accessed using the scrollbar, the mouse wheel, or the keyboard. Hold the **Shift** key while pressing the **Up** and **Down** arrow keys to scroll line-by-line or the **PageUp** and **PageDown** keys to scroll page-by-page.

Mouse tracking

When an application activates mouse tracking, mouse events are sent to the application rather than being treated as window events. This is indicated by the mouse pointer changing from an **I** shape to an arrow. Holding down **Shift** overrides mouse tracking mode and sends mouse events to the window instead, so that e.g. text can be selected and the menu can be accessed.

TERM variable

By default, the *TERM* variable for the child process is set to "xterm", so that programs that pay attention to it expect xterm keycodes and output xterm-compatible control sequences.

Shortcuts

An overview of all the keyboard shortcuts.

Window commands

- **Alt+F2**: Duplicate
- **Alt+F4**: Close
- **Alt+F8**: Reset
- **Alt+F10**: Default size
- **Alt+F11** or **Alt+Enter**: Fullscreen
- **Alt+Space**: Window menu

Scrollback

- **Shift+Up**: Line up
- **Shift+Down**: Line down
- **Shift+PgUp**: Page up
- **Shift+PgDown**: Page down
- **Shift+Home**: Top
- **Shift+End**: Bottom

Copy and paste

- **Ctrl+Ins**: Copy
- **Shift+Ins**: Paste

Font zoom

- **Ctrl+plus**: Zoom in
- **Ctrl+minus**: Zoom out
- **Ctrl+zero**: Back to configured font size

CONFIGURATION

Most settings are chosen not through command line arguments but in the graphical options dialog, which can be reached via the context menu or the window menu. Settings are stored in an INI-style configuration file that by default is located at `%.minttyrc`. This can be overridden with the **--config** command line option. Settings are written to the file whenever the **OK** or **Apply** buttons are pressed in the options dialog.

The following sections explain the settings on each pane of the options dialog. For each setting, its name in the config file is shown in parentheses, along with its default value, e.g. `Columns=80`. For multiple-choice settings, the value representing each choice in the config file is shown.

Looks

Settings affecting mintty's appearance.

Colours

Clicking on one of the buttons here opens the colour selection dialog. In the config file, colours are represented as comma-separated RGB triples with decimal 8-bit values (i.e. ranging from 0 to 255).

- **Foreground** (ForegroundColour=191,191,191)
- **Background** (BackgroundColour=0,0,0)
- **Cursor** (CursorColour=191,191,191)

Transparency (Transparency=0)

Window transparency level, with the following choices:

- **Off** (0)
- **Low** (1)
- **Medium** (2)
- **High** (3)
- **Glass** (-1)

The **Glass** option is only available on Vista and above with desktop compositing enabled. To make this reasonably usable, the glass colour needs to be set to be as dark as possible in the Windows control panel: choose *Personalize* from the desktop context menu, click on *Window Color*, turn the color intensity up to the maximum, show the color mixer, and turn the brightness down to black.

Opaque when focused (OpaqueWhenFocused=0)

Enable to make the window opaque when it is active (to avoid background distractions when working in it).

Cursor (CursorType=2)

The following cursor types are available:

- **Block** (0)
- **Underscore** (1)
- **Line** (2)

Enable cursor blinking (CursorBlinks=1)

If enabled, the cursor blinks at the rate set in the Windows Keyboard control panel.

Text

Settings controlling text display.

Font selection

Clicking on the **Select** button opens a dialog where the font and its properties can be chosen. In the config file, this corresponds to the following entries:

- **Font** (Font=Lucida Console)
- **Style** (FontIsBold=0)
- **Size** (FontHeight=10)

Smoothing (FontQuality=0)

Select the amount of font smoothing from the following choices:

- **Default** (0): Use Windows setting
- **None** (1): With all the jaggies
- **Partial** (2): Greyscale anti-aliasing
- **Full** (3): Subpixel anti-aliasing (ClearType)

Show bold as bright (BoldAsBright=1)

By default, text with the ANSI bold attribute set is displayed with increased brightness. Alternatively, it can be shown using a bold font, which tends to look better with black-on-white text.

Allow blinking (AllowBlinking=0)

ANSI text blinking is disabled by default, on the grounds that blinking text is a crime against aesthetic decency.

Locale (Locale=)

The locale setting consists of a lowercase two-letter language code such as **en** or **zh**, followed by an optional uppercase two-letter country code such as **US** or **CN**. If the country code is present, the two parts have to be separated by an underscore, for instance **en_US** or **zh_CN**. Alternatively, the language-neutral "C" locale can be selected.

If no locale is set here, which is the default, the locale and character set specified via the environment variables *LC_ALL*, *LC_CTYPE* or *LANG* are used instead.

If a locale is set, however, it will override any environment variable setting: *LC_ALL* and *LC_CTYPE* are cleared, while *LANG* is set according to the chosen locale and character set.

Character set (Charset=)

The character set to be used for encoding input and decoding output. If no locale is set, this setting is ignored.

While changing the character set immediately takes effect for text input and output, it does not affect the processes already running in mintty. This is because the environment variables of a running process cannot be changed from outside that process. Therefore mintty needs to be restarted for a character set change to take full effect.

Keys

Settings controlling keyboard behaviour.

Backspace sends ^H (BackspaceSendsBS=0)

By default, mintty sends `^?` as the keycode for the backspace key. If this option is enabled, `^H` is sent instead. This also changes the `Ctrl+Backspace` code from `^_` to `^?`.

Lone Alt sends ESC (AltSendsESC=0)

The Alt key pressed on its own can be set to send the escape character `^[`. This can save the regular trip to the upper left corner of the keyboard for *vi* users.

Ctrl+LeftAlt is AltGr (CtrlAltIsAltGr=0)

The AltGr key on non-US Windows systems is a strange beast: pressing it is synonymous with pressing the left Ctrl key and the right Alt key at the same time, and Windows programs usually treat any Ctrl+Alt combination as AltGr.

Some programs, however, chief among them Microsoft's very own Office, do not treat Ctrl+Left-Alt as AltGr, so that Ctrl+LeftAlt combinations can be used in command shortcuts even when a key has an AltGr character binding.

By default, mintty follows Office's approach, because a number of terminal programs make use of Ctrl+Alt shortcuts. The "standard" Windows behaviour can be restored by ticking the checkbox here.

The setting makes no difference for keys without AltGr key bindings (e.g. any key in the standard US layout).

Menu and fullscreen shortcuts (WindowShortcuts=1)

Checkbox for enabling the **Alt+Space** and **Alt+Enter** shortcuts.

Zoom shortcuts (ZoomShortcuts=1)

Checkbox for enabling the font zooming shortcuts **Ctrl+plus/minus/zero**.

Modifier for scrolling (ScrollMod=1)

The modifier key that needs to be pressed together with the arrow up/down or page up/down keys to access the scrollbar buffer. The default is **Shift**. The **Off** setting disables scrolling with the cursor keys.

- **Shift** (1)
- **Ctrl** (4)
- **Alt** (2)
- **Off** (0)

Mouse

Settings controlling mouse support.

Copy on select (CopyOnSelect=0)

If enabled, the region selected with the mouse is copied to the clipboard as soon as the mouse button is released, thus emulating X Window behaviour.

Clicks place cursor (ClicksPlaceCursor=0)

If enabled, the command line cursor can be placed by pressing the left mouse button. This works by sending the number of cursor keycodes needed to get to the destination.

Right click action (RightClickAction=0)

Action to take when the right mouse button is pressed.

- **Paste** (1): Paste the clipboard contents.
- **Extend** (2): Extend the selected region.
- **Show menu** (0): Show the context menu.

Default click target (ClicksTargetApp=1)

This applies to application mouse mode, i.e. when the application activates xterm-style mouse reporting. In that mode, mouse clicks can be sent either to the application to process, or to the window for the usual actions such as select and paste.

- **Window** (0)
- **Application** (1)

Modifier key for overriding default (ClickTargetMod=1)

The modifier key selected here can be used to override the default click target in application mouse mode. With the default settings, clicks are sent to the application, and Shift has to be pressed while clicking in order to trigger window actions instead.

The **Off** setting disables overriding. If the default target is set to **Window**, this effectively disables application mouse mode.

- **Shift** (1)

- **Ctrl** (4)
- **Alt** (2)
- **Off** (0)

Output

Settings for output devices other than the screen.

Printer (Printer=)

The ANSI standard defines control sequences for sending text to a printer, which are used by some terminal applications such as the mail reader **pine**. The Windows printer to send such text to can be selected here. By default, printing is disabled.

Bell The three checkboxes here determine what effects the bell character **^G** has. Taskbar highlighting, which is enabled by default, changes the colour of mintty's taskbar entry in case its window is not active already.

- **Play Sound** (BellSound=0)
- **Flash Screen** (BellFlash=0)
- **Highlight in taskbar** (BellTaskbar=1)

TERM (Term=xterm)

The TERM variable setting at mintty startup. Choices available from the dropdown list are **xterm**, **xterm-256color**, and **vt100**. This setting has no effect on mintty's terminal emulation, but it tells applications what features to expect. The **xterm-256color** setting enables 256-color mode in some applications, but may not be recognised at all by others, which is why plain **xterm** is the default.

Answerback (Answerback=)

The answerback string is sent in response to the **^E** (ENQ) character. By default, this is empty.

Window

Window properties.

Columns (Columns=80)

Default width of the window, in character cells.

Rows (Rows=24)

Default height of the window, in character cells.

Current size

Pressing this button sets the default width and height of the window to its current size.

Show scrollbar (Scrollbar=1)

Show the scrollbar for accessing the scrollbar buffer.

Enable scrollbar on alternate screen (AltScreenScroll=0)

Fullscreen applications such as editors and viewers usually switch to a second screen buffer called the *alternate screen*. This allows the screen to be restored to its original content after the application finishes.

By default, mousewheel events and the key combinations for scrolling the screen are sent to the application to enable scrolling within that application. If the option here is enabled, however, the

command line history on the primary screen can be accessed instead.

Scrollback lines (ScrollbackLines=10000)

The maximum number of lines to keep in the scrollback buffer.

Ask for exit confirmation (ConfirmExit=1)

If enabled, ask for confirmation when the close button or *Alt+F4* is pressed and the command invoked by mintty still has child processes. This is enabled by default is intended to help avoid closing programs accidentally.

KEYCODES

The Windows keyboard layout is used to translate alphanumeric and symbol key presses into characters, which means that the keyboard layout can be switched using the standard Windows mechanisms for that purpose. Also, **AltGr** combinations, dead keys, and input method editors (IMEs) are all supported.

Should the available keyboard layouts lack required features, Microsoft's **Keyboard Layout Creator** (MSKLC), available from <http://www.microsoft.com/Globaldev/tools/msklc.msp>, can be used to create custom keyboard layouts.

For other keys, mintty sends xterm keycodes as described at <http://invisible-island.net/xterm/ctlseqs/ctlseqs.html>, with a few minor changes and additions.

Caret notation is used to show control characters. See http://en.wikipedia.org/wiki/Caret_notation for an explanation.

Alt and Meta

As is customary with PC keyboards, the **Alt** key acts as the so-called **Meta** modifier. When it is held down while pressing a key or key combination, the keycode is prepended with the escape character `^`, unless noted otherwise in the keycode tables in the following sections.

Encoding the meta modifier by setting the top bit of a character instead of prefixing it with the escape character is not supported, because that does not work for character codes beyond 7-bit ASCII.

Letter keys

If the Windows keyboard layout does not have a keycode for a letter key press and the **Ctrl** key is down, a control character corresponding to the key's "virtual keycode" is sent. For keyboards with Latin scripts the virtual keycodes reflect the keys' labels, whereas for others, the virtual keys are usually arranged in the same way as on a QWERTY keyboard.

<i>Key</i>	Ctrl	Ctrl+Alt
A	<code>^A</code>	<code>^[^A</code>
B	<code>^B</code>	<code>^[^B</code>
...		
Z	<code>^Z</code>	<code>^[^Z</code>

Special keys

The keys here send the usual control characters, but there are a few mintty-specific additions that make combinations with modifier keys available as separate keycodes.

<i>Key</i>		Shift	Ctrl	C+S	Alt	A+S
Space	<i>SP</i>	<i>SP</i>	^@	^[^@	^[<i>SP</i>	^[<i>SP</i>
Enter	^M	^J	^^	^[^^	^[^M	^[^J
Back	^?	^?	^_	^[^_	^[^?	^[^?
Tab	^I	^[[Z	^[[1;5I	^[[1;6I		
Pause	^]					
Break	^\					

On most keyboards **Pause** and **Break** share a key, whereby **Ctrl** has to be pressed to get the **Break** function.

Modifier Keys

The remaining keys all use a common encoding for modifier keys. When one or more of the following modifier keys are pressed, they are encoded by adding the associated value to 1.

- **Shift**: 1
- **Alt** : 2
- **Ctrl** : 4

For example, **Shift+Ctrl** would be encoded as the character **6** (i.e. 1+1+4). The modifier code is shown as *m* in the following sections.

Number and symbol keys

In the same way as for letter keys, the Windows keyboard layout is consulted first for the number and symbol keys. Key combinations involving **Ctrl** send the following keycodes:

<i>Key</i>	modified
*	^[[1;mj
+	^[[1;mk
,	^[[1;ml
-	^[[1;mm
/	^[[1;mo
0	^[[1;mp
1	^[[1;mq
2	^[[1;mr
...	
9	^[[1;my

Cursor keys

Cursor keycodes without modifier keys depend on the terminal's "application cursor key mode", which is often used by fullscreen applications. When one or more modifier keys are pressed, the application cursor mode is ignored, but the modifier code is inserted into the keycode as shown. The **Home** and **End** keys are considered cursor keys.

<i>Key</i>		app	modified
Up	^[[A	^[OA	^[[1;mA
Down	^[[B	^[OB	^[[1;mB
Left	^[[D	^[OD	^[[1;mD
Right	^[[C	^[OC	^[[1;mC
Home	^[[H	^[OH	^[[1;mH

End $\hat{[[F}$ $\hat{[OF}$ $\hat{[[1;mF}$

Editing keys

There is no special application mode for the remaining four keys in the block of six that is usually situated above the cursor keys.

<i>Key</i>		modified
Ins	$\hat{[[2\sim}$	$\hat{[[2;m\sim}$
Del	$\hat{[[3\sim}$	$\hat{[[3;m\sim}$
PgUp	$\hat{[[5\sim}$	$\hat{[[5;m\sim}$
PgDn	$\hat{[[6\sim}$	$\hat{[[6;m\sim}$

Function keys

F1 through **F4** send numpad-style keycodes, because they emulate the four PF keys above the number pad on the VT100 terminal. The remaining function keys send codes that were introduced with the VT220 terminal.

<i>Key</i>		modified
F1	$\hat{[OP}$	$\hat{[[1;mP}$
F2	$\hat{[OQ}$	$\hat{[[1;mQ}$
F3	$\hat{[OR}$	$\hat{[[1;mR}$
F4	$\hat{[OS}$	$\hat{[[1;mS}$
F5	$\hat{[[15\sim}$	$\hat{[[15;m\sim}$
F6	$\hat{[[17\sim}$	$\hat{[[17;m\sim}$
F7	$\hat{[[18\sim}$	$\hat{[[18;m\sim}$
F8	$\hat{[[19\sim}$	$\hat{[[19;m\sim}$
F9	$\hat{[[20\sim}$	$\hat{[[20;m\sim}$
F10	$\hat{[[21\sim}$	$\hat{[[21;m\sim}$
F11	$\hat{[[23\sim}$	$\hat{[[23;m\sim}$
F12	$\hat{[[24\sim}$	$\hat{[[24;m\sim}$

Alt+Numpad

The Windows Alt+Numpad method for entering character codes is supported, whereby the **Alt** key has to be held while entering the character's Unicode codepoint. If the first digit entered is a zero, the codepoint is interpreted as octal, otherwise as decimal. The codepoint is encoded using the selected codepage before it is sent.

Mousewheel

In xterm mouse reporting modes, the mousewheel is treated as a pair of mouse buttons. However, the mousewheel can also be used for scrolling in applications such as *less* that do not support xterm mouse reporting but that do use the alternate screen. Under those circumstances, mousewheel events are encoded as cursor up/down or page up/down keys. See the cursor keycode and editing keycodes above for details.

The number of line up/down events sent per mousewheel notch depends on the relevant Windows setting on the **Wheel** tab of the **Mouse** control panel. Page up/down codes can be sent by holding down **Shift** while scrolling. The Windows wheel setting can also be set to always scroll by a whole screen at a time.

CONTROL SEQUENCES

Most of the xterm control sequences documented at <http://invisible-island.net/xterm/ctlseqs/ctlseqs.html> are supported. Please report incompatibilities or unimplemented sequences as bugs.

This section lists mintty-specific control sequences.

Application mousewheel mode

Application mousewheel mode allows the mousewheel to be distinguished from the cursor keys without enabling full xterm mouse tracking. The two sequences enable and disable it:

sequence	mode
<code>^[[?7787l</code>	normal
<code>^[[?7787h</code>	application

When application mousewheel mode is on, mousewheel events are encoded as follows:

line up	<code>^[Oa</code>
line down	<code>^[Ob</code>
page up	<code>^[[1;2a</code>
page down	<code>^[[1;2b</code>

Escape keycode

There are two settings controlling the keycode sent by the escape key.

The first controls application escape key mode, where the escape key sends a keycode that allows applications such as vim to tell it apart from the escape character appearing at the start of many other keycodes, without resorting to a timeout mechanism. This setting also applies to the Alt key when that is pressed on its own.

sequence	mode	keycode
<code>^[[?7727l</code>	normal	<code>^[</code> or <code>^\</code>
<code>^[[?7727h</code>	application	<code>^[O[</code>

When application escape key mode is off, the escape key can be configured to send `^\` instead of the standard and default `^[`. This allows the escape key to be used as one of the special keys in the terminal line settings (see `stty(1)`), which is not possible with `^[`, as that appears as the first character in many other keycodes.

sequence	keycode
<code>^[[?7728l</code>	<code>^[</code>
<code>^[[?7728h</code>	<code>^\</code>

Shortcut override mode

When shortcut override mode is on, all shortcut key combinations are sent to the application instead of triggering window commands.

sequence	override
<code>^[[?7783l</code>	off
<code>^[[?7783h</code>	on

Ambiguous width reporting

Applications can ask to be notified when the width of the so-called "ambiguous width" character category changes due to the user changing font.

sequence	reporting
<code>^[[?7700l</code>	disabled
<code>^[[?7700h</code>	enabled

When enabled, `^[[1W` is sent when changing to an "ambiguous narrow" font and `^[[2W` is sent when changing to an "ambiguous wide" font.

Font size

These OSC sequences can be used to set and query font size:

sequence	font size
<code>^[]7770 ; ? ^G</code>	query
<code>^[]7770 ; num ^G</code>	set to <i>num</i>
<code>^[]7770 ; +num ^G</code>	increase by <i>num</i>
<code>^[]7770 ; -num ^G</code>	decrease by <i>num</i>
<code>^[]7770 ; ^G</code>	default

As usual, OSC sequences can also be terminated with `^[\ (ST)` instead of `^G`. When the font size is queried, a sequence that would restore the current size is sent, terminated with ST: `^[]7700 ; num ^[\`.

Locale

The locale and charset used by the terminal can be queried or changed using these sequences:

sequence	locale
<code>^[]7776 ; ? ^G</code>	query
<code>^[]7776 ; loc ^G</code>	set to <i>loc</i>
<code>^[]7776 ; ^G</code>	default

The locale string used here should take the same format as in the locale environment variables such as *LANG*. When the locale is queried, a sequence that would set the current locale is sent, e.g.

`^[]7776 ; C.UTF-8 ^[\`. An empty *loc* string selects the locale configured in the options or the environment.

Cursor style

The VT510 DECCSUSR sequence can be used to control cursor shape and blinking.

```
^[[ arg SP q
```

<i>arg</i>	shape	blinking
0	default	default
1	block	yes
2	block	no
3	underscore	yes
4	underscore	no
5	line	yes
6	line	no

TIPS

A few tips on setting up mintty and other programs.

Shortcuts

The Cygwin package for mintty installs a shortcut in the Windows start menu under *All Programs/Cygwin*. It starts mintty with a '-' as its only argument, which tells it to invoke the user's default shell as a login shell.

Shortcuts are also a convenient way to start mintty with additional options and different commands. For example, shortcuts for access to remote machines can be created by invoking *ssh*. The command simply needs to be appended to the target field of the shortcut (in the shortcut's properties):

```
Target: C:\Cygwin\bin\mintty.exe ssh server
```

The working directory for the session can be set in the "Start In:" field. (But note that the bash login profile *cd*'s to the user's home directory.) Another convenient feature of shortcuts is the ability to assign global shortcut keys.

Cygwin provides the **mkshortcut** utility for creating shortcuts from the command line. See its manual page for details.

Starting mintty from folder context menus

Cygwin's **chere** package can be used to create a folder context menu item for mintty in Windows Explorer. This allows one to right click on a folder and open a shell in that folder.

The following command will create a "Bash Prompt Here" for the current user. See *chere(1)* for all the options.

```
chere -i -c -t mintty
```

Starting mintty from a batch file

In order to start mintty from a batch file it needs to be invoked through the *start* command. This avoids the batch file's console window staying open while mintty is running. For example:

```
start mintty -
```

Environment variables

Unfortunately Windows shortcuts do not allow the setting of environment variables. Variables can be set globally though via a button on the **Advanced** tab of the **System Properties**. Those can be reached by right-clicking on **Computer**, selecting **Properties**, then **Advanced System Settings**.

Alternatively, global variables can be set using the *setx* command line utility. This comes pre-installed with some versions of Windows but is also available as part of the freely downloadable **Windows 2003 Resource Kit Tools**.

Another way to set variables for the program to be run in **mintty** is by invoking it through the **env** command, e.g.:

```
mintty env DISPLAY=:0 ssh -X server
```

The CYGWIN variable

The **CYGWIN** environment variable is used to control a number of settings for the Cygwin runtime system. Many of them apply to the Cygwin console only, but others can be useful with any Cygwin process. See <http://www.cygwin.com/cygwin-ug-net/using-cygwinenv.html> for details.

Changing the ANSI colours

A number of settings can be controlled through terminal control sequences, including the colour values for the 16 ANSI colours. Here is the xterm sequence for this, whereby *num* stands for the ANSI number and *rrggbb* stands for a hexadecimal RGB colour value.

```
^[ ]4;num;#rrggbb^G
```

The **-e** option to the **echo** command is useful for emitting control sequences. For example, to turn yellow (colour 3) up to its full brightness:

```
echo -e "\e]4;3;#FFFF00\a"
```

Sequences such as this can be included in scripts or on the **mintty** command line with the help of **sh -c**.

Terminal line settings

Terminal line settings can be viewed or changed with the **stty** utility, which is installed as part of Cygwin's core utilities package. Among other things, it can set the control characters used for generating signals or editing an input line.

See the **stty** man page for all the details, but here are a few examples. The commands can be included in shell startup files to make them permanent.

To change the key for deleting a whole word from **Ctrl+W** to **Ctrl+Backspace**:

```
stty werase '^_'
```

To use **Ctrl+Enter** instead of **Ctrl+D** for end of file:

```
stty eof '^^\n'
```

To use **Pause** and **Break** instead of **Ctrl+Z** and **Ctrl+C** for suspending or interrupting a process, and to also disable the stackdump-producing SIGQUIT:

```
stty susp '^]' swtch '^]' intr '^^\n' quit '^_'
```

With these settings, the **Esc** key can also be used to interrupt processes by setting its keycode to `^\`

```
echo -e "\[?7728h"
```

The standard escape character `^[` cannot be used for that purpose because it appears as the first character in many keycodes.

Readline configuration

Keyboard input for the **bash** shell and other program that use the **readline** library can be configured with the so-called *inputrc* file. Unless overridden by setting the *INPUTRC* variable, this is located at `~/inputrc`.

It consists of bindings of keycodes to readline commands, whereby comments start with a hash character. The file format is explained fully in the bash manual.

Anyone used to Windows key combinations for editing text might find the following bindings useful:

```
# Ctrl+Left/Right to move by whole words
"\e[1;5D": backward-word
"\e[1;5C": forward-word

# Ctrl+Backspace/Delete to delete whole words
"\C-_" : backward-kill-word
"\e[3;5~": kill-word

# Ctrl+Shift+Backspace/Delete to delete to start/end of the line
"\e\C-_" : backward-kill-line
"\e[3;6~": kill-line

# Alt-Backspace for undo
"\e\d": undo
```

Finally, a couple of bindings for convenient searching of the command history. Just enter the first few characters of a previous command and press **Ctrl-Up** to look it up.

```
# Ctrl-Up/Down for searching command history
"\e[1;5A": history-search-backward
"\e[1;5B": history-search-forward
```

Enabling non-ASCII input in bash and readline

By default, **readline** treats the uppermost bit of input and output characters as the "meta" flag. The following settings in `~/inputrc` will disable that and thereby allow the use of characters outside the 7-bit ASCII character set:

```
set input-meta on
set convert-meta off
set output-meta on
```

Mode-dependent cursor in vim

Using the control sequences for cursor style and application escape key mode, the **vim** editor can be configured to change cursor depending on mode. For example, with the following lines in `.vimrc`, vim will show a block cursor in normal mode and a line cursor in insert mode:

```
let &t_ti.=" \e[1 q\e[?7727h"
let &t_SI.=" \e[5 q"
let &t_EI.=" \e[1 q"
let &t_te.=" \e[0 q\e[?7727l"
noremap <Esc>O[ <Esc>
noremap! <Esc>O[ <Esc>
```

Without application escape key mode, it would take a second for the cursor to change after pressing Esc to leave insert mode. (That time can otherwise be reduced using vim's `timeoutlen` setting, at the risk of keycodes not being recognised in some circumstances.)

LIMITATIONS

Console Issue

Mintty is not a full replacement for the Windows console window that Cygwin uses by default. Like xterm and rxvt, mintty communicates with the child process through a pseudo terminal device, which Cygwin emulates using Windows pipes. This means that native Windows command line programs started in mintty see a pipe rather than a console device. As a consequence, such programs often disable interactive input. Also, direct calls to low-level Win32 console functions will fail. Programs that access the console as a file should be fine though.

Termcap/terminfo

Mintty does not have its own *termcap* or *terminfo* entries; instead, it simply pretends to be an xterm.

Missing xterm features

Mintty is nowhere near as configurable as xterm, and its keycodes are fixed according to xterm's PC-style keyboard behaviour (albeit with a number of mintty-specific extensions). Also, there is no Tektronix 4014 emulation or mouse highlighting mode.

SEE ALSO

bash(1), *env(1)*, *echo(1)*, *stty(1)*, *script(1)*, *mkshortcut(1)*, *chere(1)*, *lesskey(1)*, *vim(1)*

<http://invisible-island.net/xterm/ctlseqs/ctlseqs.html>

<http://vt100.net>

ACKNOWLEDGEMENTS

Mintty is based on PuTTY version 0.60 by Simon Tatham and contributors, so big thanks to everyone involved. Thanks also to KDE's Oxygen team for the program icon.

COPYRIGHT

Copyright (C) 2008-09 Andy Koppe.

Mintty is released under the terms of the the *GNU General Public License* version 3 or later. See <http://gnu.org/licenses/gpl/html> for the license text.

There is NO WARRANTY, to the extent permitted by law.

CONTACT

Please report bugs or suggest enhancements via the issue tracker at <http://code.google.com/p/mintty/issues>. Questions can be sent to the discussion group at <http://groups.google.com/group/mintty-discuss> or the Cygwin mailing list at cygwin@cygwin.com.

AUTHOR

This manual page was written by Andy Koppe with much appreciated help from Lee D. Rothstein.