NAME

mintty - Cygwin terminal emulator

SYNOPSIS

mintty [*OPTION*]... [**-e**] [**-** | *COMMAND* [*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 to be running.

If a command is supplied on the command line, MinTTY executes that with any arguments given. If a single dash is supplied instead of a command, the current user's default shell (as defined in */etc/passwd*) is invoked as a login shell. Otherwise, it is invoked as a non-login shell. The command can be preceded by **-e** (for execute), but that is not required.

OPTIONS

MinTTY accepts the standard GNU option formats, with single dashes introducing short options and double dashes introducing long options, e.g. **-o arg**, **--option arg**, and **--option=arg**.

-c, **--config**=*FILENAME*
: Use the specified configuration file instead of the default *~/.minttyrc*.

-p, **--pos**=*X*,*Y*
: Open the window with its top left corner at the specified coordinates.

-s, **--size**=*COLS*,*ROWS*
: Set the number of screen columns and rows, overriding any config file settings.

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

-v, **--version**
: Print version information and exit.

-h, **--help**
: Display a brief help message and exit.

USAGE

MinTTY aims to adhere to both Windows and Unix conventions, but where they conflict the Windows conventions take precedence, at least in the default configuration.

Menu

MinTTY's 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.

Options dialog

The configuration can be changed in a graphical dialog, which can be opened by selecting **Options...** in MinTTY's context menu or window menu. (The window menu can be opened by clicking on the program icon or pressing Alt+Space.)

See the **CONFIGURATION** section for details.

Copy & paste

As usual in Windows, screen contents can be selected by holding down the left mouse button and dragging the mouse and the current selection can be extended by holding down **Shift** or **Ctrl** while left-clicking. Double-clicking or triple-clicking selects a whole word or line.

In the default configuration, selected text has to be explicitly copied to the clipboard using either the **Copy** menu command, the **Ctrl+Insert** keyboard shortcut, or the middle mouse button combined with **Shift**. (See the **Copy on select** setting for X-like copy behaviour.)

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 **Ctrl+Insert** 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 also be dropped into the MinTTY window.

Fullscreen

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

Scrolling

MinTTY has a scrollback buffer that can hold up to 65535 lines. This 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. The modifier key for scrolling can be changed in the options dialog.

Scrolling into the scrollback buffer is not available with programs such as pagers or editors that use the the terminal's "alternate screen".

Mouse tracking

When an application activates mouse tracking, mouse events are sent to the application rather than processed by MinTTY. 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 MinTTY instead, so that e.g. text can be selected and the menu can be accessed. (The modifier key for overriding mouse tracking can be changed in the options dialog.)

TERM variable
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.

CONFIGURATION

Most MinTTY settings are chosen not through command line arguments but in its graphical options dialog, which can be reached via the context menu or the window menu. Settings are stored in a 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** button is 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.

Window
Window properties.

Columns (Columns=80)
Width of the window in character cells.

Rows (Rows=24)
Height of the window in character cells.

Transparency (Transparency=0)
Window transparency level, with the following choices:

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

Disable transparency when active (OpaqueWhenFocused=0)
Make the window opaque when it is active (to avoid background distractions when working in it).

Display scrollbar (Scrollbar=1)
Show the scrollbar for accessing the scrollback buffer on the right of the window.

Enable Alt+key shortcuts (WindowShortcuts=1)
Controls shortcuts for window commands. When disabled, these combinations send their regular keycodes to the application.

- **Alt+Space**: Menu
- **Alt+Enter**: Fullscreen
- **Alt+F2**: Duplicate
- **Alt+F4**: Close

Looks
Settings for changing 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)

Show bold text as bright (BoldAsBright=1)
If selected, text with the ANSI bold attribute set is displayed with increased brightness. Otherwise, it is shown with a bold font, which tends to look better with black-on-white text.

Allow text blinking (AllowBlinking=1)
ANSI text blinking can be disabled here, as it can be rather annoying.

Cursor (CursorType=2)
The following cursor types are available:

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

Enable cursor blinking (CursorBlinks=1)
If enabled, the cursor blinks at the rate set in Windows' keyboard control panel.

Font
Settings controlling text display.

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

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

Smoothing (FontQuality=0)
Select the type of font smoothing, if any, from the following choices:

- **System Default** (0)
- **None** (2)
- **Antialiased** (1)
- **ClearType** (3)

Codepage (Codepage=ISO-8859-1:1998 (Latin-1, West Europe))
The codepage used for encoding input and decoding output. Select **UTF-8** for 8-bit Unicode.

Keys
Settings controlling keyboard behaviour.

Backspace keycode (BackspaceSendsDEL=0)
The character to be sent by the backspace key. The default is **^H**, because that is the default across Cygwin, but **^?** (DEL) can be used instead to free up Ctrl+H for other purposes, e.g. as the help key in Emacs.

- **^H** (0)
- **^?** (1)

Escape keycode (EscapeSendsFS=0)
The character to be sent by the escape key. The default is the standard escape character **^[**, but the character ^\ can be used instead, thereby allowing the escape key to be used as one of the special keys in the terminal line settings (see stty(1)). This is impractical with **^[**, as that appears as the first character in multi-character keycodes.

- **^[** (0)
- **^\fP (1)**

Alt key on its own sends ^[(AltSendsESC=0)
The Alt key pressed on its own can be set to send the escape character **^[**. This can be particularly useful when the escape key is set to send ^\ instead.

Modifier key 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 scrollback buffer.

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

Mouse
Settings controlling mouse support.

Right click action (RightClickAction=0)
Action to take when clicking the right mouse button.

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

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.

Default click target (ClickTargetsApp=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: select, extend, paste, show menu.

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

Modifier key 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.

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

Output

Settings for output devices other than the terminal 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 action (BellType=1)
The action to take when the application sends the bell character **^G**.

- **None** (0)
- **System sound** (1)
- **Flash window** (2)

KEYCODES

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

The Windows keyboard layout yields Unicode codepoints, which are encoded using the **Codepage** selected in MinTTY's configuration before sending them to the application. (The UTF-8 codepage can be selected for full Unicode input support.)

Should the available keyboard layouts lack required features, Microsoft's **Keyboard Layout Creator** (MSKLC), available from *http://www.microsoft.com/Globaldev/tools/msklc.mspx*, 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.

Letter keys

If the Windows keyboard layout does not have a keycode for a letter key press and the **Ctrl** key is down, MinTTY sends a control character. The character sent corresponds to the key's "virtual keycode". For keyboards with Latin scripts the virtual keycodes reflect the keys' labels, whereas for others, the virtual keys are usually laid out the same as on the US keyboard.

Key	**Ctrl**	**Ctrl+Shift/Alt**
A	^A	^[^A
B	^B	^[^B
...		

Z	^Z	^[^Z

Number and symbol keys

In the same way as for letter keys, the Windows keyboard layout is consulted first for number and symbol keys. If that comes back empty, and **Ctrl** is down, the keycodes below are sent.

Unlike xterm, MinTTY ignores VT100 "application keypad mode". Instead, it relies on the state of **NumLock** to decide how to handle number pad keys. As usual on Windows, when **NumLock** is off, the number pad keys are treated as arrow and editing keys, and when it is on, they are treated as number and symbol keys. Application keypad codes can still be sent though, by holding down **Ctrl** while **NumLock** is on.

Furthermore, the number keys as well as the comma, period, plus and minus keys on the main part of the keyboard also send application keypad codes when pressed simultaneously with **Ctrl**. This makes those keycodes more accessible to laptop users and more useful as application shortcuts.

Finally, the keycodes can be modified by holding **Shift** or **Alt** as well as **Ctrl**.

Key	**Ctrl**	**Ctrl+Shift/Alt**
*****	^[Oj	^[[j
+	^[Ok	^[[k
,	^[Ol	^[[l
-	^[Om	^[[m
.	^[On	^[[n
/	^[Oo	^[[o
0	^[Op	^[[p
1	^[Oq	^[[q
...		
9	^[Oy	^[[y

Some of the symbol keys send control characters when pressed together with **Ctrl**. These are the characters between **^Z** (ASCII 26) and space (32). Their positions on the keyboard are hard-coded based on the US keyboard layout.

Key	**Ctrl**	**Ctrl+Shift/Alt**
[{	^[	^[^[
]}	^]	^[^]
|	^\	^[^\
'"	^^	^[^^
/?	^_	^[^_

Control 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.

Key		**Shift**	**Crtl**	**C+S**	**Alt**	**A+S**
Space	*sp*	*sp*	^@	^[^@	^[*sp*	^[*sp*
Enter	^M	^J	^^	^[^^	^[^M	^[J
Back (^H)	^H	^H	^?	^[^?	^[^H	^[*sp*
Back (^?)	^?	^?	^_	^[^_	^[^?	^[*sp*
Tab	^I	^[[Z	^[Oz	^[OZ		
Esc (^[)	^[	^]				
Esc (^\)	^\	^]				

Pause	^]	^[^]
Break	^\	^[^\

The **Back** and **Esc** keycodes can be configured in the options dialog, which is why different keycodes depending on those settings are shown. 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.

Cursor keys

Cursor keycodes without modifier keys depend on the terminal's "application cursor mode", which is used by fullscreen applications such as editors and pagers. 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.

Key		**app**	**modified**
Up	^[[A	^[OA	^[[1;*m*A
Down	^[[B	^[OB	^[[1;*m*B
Left	^[[D	^[OD	^[[1;*m*D
Right	^[[C	^[OC	^[[1;*m*C
Home	^[[H	^[OH	^[[1;*m*H
End	^[[F	^[OF	^[[1;*m*F

Editing keys

There is no special application mode for the editing keys in the block of six that is usually situated above the cursor keys, but modifiers can be applied.

Key		**modified**
Insert	^[[2~	^[[2;*m*~
Delete	^[[3~	^[[3;*m*~
PgUp	^[[5~	^[[5;*m*~
PgDn	^[[6~	^[[6;*m*~

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.

Key		**modified**
F1	^[OP	^[[1;*m*P

F2	^[OQ	^[[1;*m*Q
F3	^[OR	^[[1;*m*R
F4	^[OS	^[[1;*m*S
F5	^[[15~	^[[15;*m*~
F6	^[[17~	^[[17;*m*~
F7	^[[18~	^[[18;*m*~
F8	^[[19~	^[[19;*m*~
F9	^[[20~	^[[20;*m*~
F10	^[[21~	^[[21;*m*~
F11	^[[23~	^[[23;*m*~
F12	^[[24~	^[[24;*m*~

Mousewheel

In xterm mouse reporting modes, the mousewheel is treated is 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 alternatescreen. Under those circumstances, mousewheel events are encoded as arrow up/down or page/up down keys, combined with the **Modifier key for scrolling** as selected on the **Keys** page of the options dialog.

line up	^[[1;*m*A
line down	^[[1;*m*B
page up	^[[5;*m*~
page down	^[[6;*m*~

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.

TIPS

A few tips on MinTTY use.

Shortcuts

The mintty Cygwin package 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 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 using the shell's **-c** option, which allows a shell command to be passed as a string argument, e.g.:

```
mintty sh -c "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** (assuming the **Backspace** keycode is set to **^H**):

```
stty werase '^?'
```

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

```
stty eof '^^'
```

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 '^´ quit '^-'
```

With these settings, the **Esc** key can also be used to interrupt processes by setting its keycode to **^**. The standard escape character **^[** cannot be used for that purpose because it appears as the first character in many other 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
"\d": backward-kill-word
"\e[3;5~": kill-word

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

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

Mousewheel scrolling for less

No, this is not some sort of special offer, but a tip on how to enable mousewheel scrolling in the pager program **less**.

Key bindings for **less** can be specified in the text file *~/.lesskey*. Before the bindings can be used, they have be translated into the binary file *~/.less* using the **lesskey** tool (which probably saves about 0.0042 seconds

when starting **less**). See **lesskey**(1) for details.

Here are the lesskey lines needed for mousewheel support, assuming the scroll modifier key is set to the default *Shift*. For **Alt** or **Ctrl**, replace the **2**s in the keycodes with **3**s or **5**s.

```
\e[1;2A back-line
\e[1;2B forw-line
\e[5;2B back-screen
\e[6;2B forw-screen
```

LIMITATIONS

Console Issue

MinTTY is not a complete replacement for the **Cygwin** console, which is based on the Windows command prompt (*cmd.exe*). 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, interactive input often does not work correctly, and direct calls to Win32 console functions will fail. Programs that only output text are usually 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). Neither Tektronix 4014 emulation nor mouse highlighting mode are supported.

SEE ALSO

stty(1), *terminfo*(5), *bash*(1), *ssh*(1), *echo*(1), *lesskey*(1), *mkshortcut*(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

CONTACT

Please report bugs or suggest enhancements via the MinTTY issue tracker at *http://mintty.google-code.com/issues*. Questions can be directed to the MinTTY 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.