bash, howto, commandlineediting, keyboard, shortcut

Fix Me!

Working with Bash Keyboard Shortcuts

Sources

- https://ss64.com/bash/syntax-keyboard.html
- https://www.howtogeek.com/howto/ubuntu/keyboard-shortcuts-for-bash-command-shell-for-ubuntu-debian-suse-redhat-linux-etc/

The Basics

Bash is the default command-line shell on most Linux distributions and included in macOS. On Windows you can install a Linux-based bash environment (WSL, Cygwin).

The bash shell features a wide variety of keyboard shortcuts you can use. These will work in bash on any operating system. Some of them may not work if you're accessing bash remotely through an SSH or telnet session, depending on how you have your keys mapped.

Shortcut	Command
CTRL+L	Clear the screen.
CTRL+W	Delete the word starting at cursor.
CTRL+U	Clear the line i.e. Delete the all words from command line.
⇧, ⇩	Recall commands (see command history).
TAB	Auto-complete files, directory, command names and much more.
CTRL+R	Search through previously used commands (see command history)
CTRL+C	Cancel currently running commands.
CTRL+T	Swap the last two characters before the cursor.
ESC+T	Swap the last two words before the cursor.

TAB completion

Tab completion is a very useful bash feature. While typing a *file name*, *directory name*, or *command*, press TAB and bash will automatically complete what you're typing, if possible. If not, bash will show you various possible matches and you can continue typing and pressing TAB to finish typing.

Shortcut	Command
TAB	Automatically complete the file name, directory name, or command you're typing.

For example: if you have a file named *really_long_file_name* in */home/user/* and it's the only file name starting with "**r**" in that directory, you can type */home/user/r*, press TAB, and bash will automatically fill in */home/user/really_long_file_name* for you. If you have multiple files or directories starting with "**r**", bash will inform you of your possibilities. You can start typing one of them and press TAB to continue.

Moving the cursor

Use the following shortcuts to quickly move the cursor around the current line while typing a command.

Shortcut	Command
Ctrl+A, POS1	Go to the beginning of the line (Home)
Ctrl+E, END	Go to the End of the line (End)
ALT+B	Back (left) one word
ALT+F	Forward (right) one word
Ctrl+F	Forward one character
Ctrl+B	Backward one character
Ctrl+Xx	(double tap 'X') Move between the beginning of the line and the current position of the cursor. This allows you to press Ctrl+Xx to return to the start of the line, change something, and then press Ctrl+Xx to go back to your original cursor position. To use this shortcut, hold the Ctrl and tap X twice.

Editing

Use the following shortcuts to quickly delete characters, fix typos and undo your key presses. Bash also includes some basic cut-and-paste features and is able to convert characters to upper or lower case.

Shortcut	Command
CTRL+L	Clear the Screen, similar to the clear command
ALT+BACKSPACE	Delete the Word before the cursor.
CTRL+D	Delete character under the cursor
ALT+D	Delete from actual cursor position to the end of word
CTRL+H	Delete character before the cursor (Backspace)
CTRL+W	Cut the Word before the cursor to the clipboard.
CTRL+K	Cut the Line after the cursor to the clipboard.
CTRL+U	Cut/delete the Line before the cursor to the clipboard.
ALT+T	Swap current word with previous
CTRL+T	Swap the last two characters before the cursor (typo).
ESC+T	Swap the last two words before the cursor.
CTRL+Y	Paste the last thing to be cut (yank)
ALT+U	Capitalize every character from the cursor to the end of the current word, converting the characters to upper case.
ALT+L	Uncapitalize every character from the cursor to the end of the current word, converting the characters to lower case.
ALT+C	Capitalize the character under the cursor. Your cursor will move to the end of the current word.
ALT+R	Cancel the changes and put back the line as it was in the history (revert).
CTRL+_	Undo

Special keys: Tab, Backspace, Enter, Esc

Text Terminals send characters (bytes), not key strokes. Special keys such as **Tab**, **Backspace**, **Enter** and **Esc** are encoded as control characters. Control characters are not printable, they display in the terminal as ^ and are intended to have an effect on applications.

Keys	Control Character
Ctrl+I	TAB
Ctrl+J	Newline
Ctrl+M	ENTER
Ctrl+[	ESC

Many terminals will also send control characters for keys in the digit row:

Shortcut	Result
CTRL+2	^@
CTRL+3	^[ESCAPE
CTRL+4	^\
CTRL+5	^]
CTRL+6	^^
CTRL+7	^_ UNDO
CTRL+8	^? Backward-delete-char

CTRL+V tells the terminal to not interpret the following character, so CTRL+V CTRL+I will display a **TAB** character, similarly CTRL+V ENTER will display the escape sequence for the Enter key: **^M**.

Working With Your Command History

You can quickly scroll through your recent commands, which are stored in your user account's bash history file.

Shortcut	Command
CTRL+R	This will enter **Bash recall mode** which you can use to search for commands you've previously run. It recalls the last command matching the characters you provide. Press CTRL+R and start typing to search your bash history (*~/.bash_history*) for a command.
CTRL+O	Execute the command found via Ctrl+R or Ctrl+S
CTRL+G	Leave **Bash recall mode** without running a command.
CTRL+P or ⇧	Go to the previous command in the command history. Press the shortcut multiple times to walk back through the history.

Shortcut Command	
CTRL+N or ⇩	Go to the next command in the command history. Press the shortcut multiple times to walk forward through the history.
ALT+R	Revert any changes to a command you've pulled from your history if you've edited it.
CTRL+S	Go back to the next most recent command. (beware to not execute it from a terminal because this will also launch its **XOFF**).
!!	Repeat last command
!abc	Run last command starting with abc
!abc:p	Print last command starting with abc
!$	Last argument of previous command
ALT+.	Last argument of previous command
!*	All arguments of previous command
^abc^def	Run previous command, replacing abc with def

Process control

Use the following shortcuts to manage running processes.

Shortcut	Command
CTRL+C	Interrupt (kill) the current foreground process running in in the terminal. This sends the **SIGINT signal** to the process, which is technically just a request—most processes will honor it, but some may ignore it.
CTRL+D	Close the bash shell. This sends an **EOF** (End-of-file) marker to bash, and bash exits when it receives this marker. This is similar to running the exit command.
CTRL+Z	Suspend the current foreground process running in bash. This sends the **SIGTSTP signal** to the process. To return the process to the foreground later, use the **fg** *process_name* command.

Controlling the screen

The following shortcuts allow you to control what appears on the screen.

Shortcut	Command
CTRL+L	Clear the screen. This is similar to running the "clear" command.
CTRL+S	Stop all output to the screen. This is particularly useful when running commands with a lot of long, verbose output, but you don't want to stop the command itself with
CTRL+Q	Resume output to the screen after stopping it with Ctrl+Shift.

Emacs mode vs Vi Mode

The above instructions assume you're using the **default keyboard shortcut configuration** in bash. By **default**, bash uses **emacs-style** keys. If you're more used to the **vi text editor**, you can switch to **vi-style** keyboard shortcuts.

The following command will put bash into **vi mode**:

```
set -o vi
```

The following command will put bash back into the **default emacs mod**e:

```
set -o emacs
```

> "...emacs, which might be thought of as a thermonuclear word processor" ~ Emacs vs. Vi Wiki

Related

- fg - Bring a command to the foreground.
- vi editor - A one page reference to the vi editor.
- *~./.bash_history* - Text file with command history.
- Terminals Are Weird - How and why of terminal keybindings.
- Equivalent Windows Keyboard shortcuts