To enable the root account, login to your raspberry pi, then type "**sudo -i**" and hit Return / Enter. (you'll need to type your password in).  
Then type "**passwd root**" and hit Return / Enter.  
You will be asked to type a password, then confirm it.  
It will say that the password was successfully changed.  
Now you need to completely log out of your Raspberry Pi (Restarting it works as well :D), then login with:  
Username: **root**  
Password: **Whatever you just set**

**--**

**Ip addr show**

**Route**

**startx --- to windows mode**

**/etc/ssh**

[host](http://www.computerhope.com/unix/host.htm)  
[ifconfig](http://www.computerhope.com/unix/uifconfi.htm)  
[netstat](http://www.computerhope.com/unix/unetstat.htm)  
[rpcinfo](http://www.computerhope.com/unix/urpcinfo.htm)  
[traceroute](http://www.computerhope.com/unix/utracero.htm)

|  |
| --- |
| **Basic "vi" features** |

One edits a file in vi by issuing the command: vi file-to-edit.txt

The vi editor has three modes, command mode, insert mode and command line mode.

1. **Command mode:** letters or sequence of letters interactively command vi. Commands are case sensitive. The ESC key can end a command.
2. **Insert mode:** Text is inserted. The ESC key ends insert mode and returns you to command mode. One can enter insert mode with the "i" (insert), "a" (insert after), "A" (insert at end of line), "o" (open new line after current line) or "O" (Open line above current line) commands.
3. **Command line mode:** One enters this mode by typing ":" which puts the command line entry at the foot of the screen.

Partial list of interactive commands:

**Cursor Movement:**

|  |  |
| --- | --- |
| **Keystrokes** | **Action** |
| h/j/k/l | Move cursor left/down/up/right |
| spacebar | Move cursor right one space |
| -/+ | Move cursor down/up in first column |
| ctrl-d n ctrl-d | Scroll **d**own one half of a screen. Set scroll to "n" lines. New default set for half screen. |
| ctrl-u n ctrl-u | Scroll **u**p one half of a screen Set scroll to "n" lines. New default set for half screen. |
| ctrl-f n ctrl-f | Scroll **f**orward one screen Scroll forward "n" screen |
| ctrl-b n ctrl-b | Scroll **b**ack one screen Scroll back "n" screen |
| ctrl-y n ctrl-y | Scroll forward one line Scroll forward "n" lines |
| ctrl-e n ctrl-e | Scroll back one line Scroll back "n" lines |
| M (shift-m) | Move cursor to middle of page |
| H (shift-h) | Move cursor to top of page |
| L (shift-l) | Move cursor to bottom of page |
| W w 5w | Move cursor a **word** at a time (white space delimited) Move cursor a word at a time (first non-alphanumeric) Move cursor ahead 5 words |
| B b 5b | Move cursor **back** a word at a time (white space delimited) Move cursor back a word at a time (first non-alphanumeric) Move cursor back 5 words |
| E e 5e | Move cursor to **end** of word (white space delimited) Move cursor to end of word (first non-alphanumeric) Move cursor ahead to the end of the 5th word |
| 0 (zero) | Move cursor to beginning of line |
| :30 | Move cursor to line thirty |
| $ | Move cursor to end of line |
| ) | Move cursor to beginning of next sentence (delimeted by ".", "?" or "!") |
| ( | Move cursor to beginning of current sentence |
| } | Move cursor to beginning of next paragraph (delimeted by blank line or nroff macros: .IP, .LP, .PP, .QP, .P, .LI and .bp) Also see "set paragraphs" to define a paragraph. |
| { | Move cursor to beginning of current paragraph |
| ]] | Move cursor to beginning of next section (delimeted by nroff macros: .NH, .SH, .H, .HU). Also see "set sections" to define a section. |
| [[ | Move cursor to beginning of current section |
| G | Move cursor to end of file |
| % | Move cursor to the matching bracket. Place cursor on {}[]() and type "%".  Use the [matchit](http://www.vim.org/scripts/script.php?script_id=39) or [xmledit](http://www.vim.org/scripts/script.php?script_id=301) plug-in to extend this capability to XML/XHTML tags. |
| '. | Move cursor to previously modified line. |
| m ma | Mark the line on which the cursor resides. Marking requires an identifier. Mark the line as identified by the letter "a" by marking with keystroke "ma" |
| 'a | Move cursor to line mark "a" generated by marking with keystroke "ma" |
| 'A | Move cursor to line mark "A" (global between buffers) generated by marking with keystroke "mA" |
| ]' | Move cursor to next lower case mark. |
| [' | Move cursor to previous lower case mark. |

**Editing Commands:**

|  |  |
| --- | --- |
| **Keystrokes** | **Action** |
| i | Insert at cursor. Puts you in insert mode. Must use esc key to terminate insert mode. |
| I | Insert before the cursor. Puts you in insert mode. Must use esc key to terminate insert mode. |
| a | Append after cursor. Puts you in insert mode. Must use esc key to terminate insert mode. |
| A | Append at end of line. Puts you in insert mode. Must use esc key to terminate insert mode. |
| o | Open a new line below the current cursor position. Also puts you in insert mode. Must use esc key to terminate insert mode. |
| O | Open a new line above the current line. Also puts you in insert mode. Must use esc key to terminate insert mode. |
| ESC | Terminate insert mode. Terminates most other modes as well. |
| u | Undo last change |
| U | Undo all changes to entire line |
| dd 3dd | Delete line (stored in local buffer) Delete 3 lines (stored in local buffer). |
| D | Delete contents of line after cursor |
| C | Delete contents of line after cursor and insert new text. Press esc key to end insertion. |
| dw 4dw d) d$ d- dfx d'x 'ad'b d/cat | Delete word Delete 4 words Delete to end of sentence Delete all characters from cursor to end of line Delete current and previous line Delete from cursor to first occurance of the letter "x" Delete from the current line to the line marked with the identifier "x" Delete from the line of mark "a" to the line marked "b". Delete all characters from the cursor to the next occurance of (but not including) "cat" |
| cw c) c$ | Change word Change sentence Change from cursor to end of line (See "d" delete above for other variations) |
| x | Delete character at cursor |
| X | Delete character before cursor |
| Y or yy | Yank (copy) current line into "unnamed" storage buffer. |
| p | Paste unnamed storage buffer after current line. |
| P | Paste unnamed storage buffer before current line. |
| r | Replace character |
| R | Overwrite characters from cursor onward |
| s | Substitute one character under cursor continue to insert |
| S | Substitute entire line and begin to insert at beginning of line |
| J | Join current and following line into one line |
| ~ | Change case of individual character |
| ctrl-a ctrl-x | Increment number under the cursor. Decrement number under the cursor. |
| . | repeat last command action. |

**Control Characters:** Note that to enter control characters while in insert mode, prefix the the control character with "ctrl-v" and then type the control character (ex. Carriage control: ctrl-M, Form feed: ctrl-L, Backspace: ctrl-H, Delete: ctrl-P, ...). Each control character must first be preceeded by ctrl-v while in insert mode.

**Delete/Restore Buffers:** Each time you delete or yank a line, it is stored in a local buffer and can be recalled and pasted. See "vi line buffers" examples below.

**Search Commands:**

|  |  |
| --- | --- |
| **Keystrokes** | **Action** |
| /*search\_string*{CR} | Search for *search\_string* |
| ?*search\_string*{CR} | Search backwards (up in file) for *search\_string* |
| /\<*search\_string*\>{CR} | Search for *search\_word* Ex: /\<s\> Search for variable "s" but ignore declaration "string" or words containing "s". This will find "string s;", "s = fn(x);", "x = fn(s);", etc |
| n | Find **n**ext occurrence of search\_word |
| N | Find previous occurrence of search\_word |
| fx nfx ; | Move cursor to **f**irst occurance of letter "x" after the cursor but in the same line Move cursor to "n"th occurance of letter "x" in line Go to next occurance in line |
| Fx nFx ; | Move cursor backwards to next occurance of letter "x" in line Move cursor backwards to "n"th occurance of letter "x" in line Go to previous occurance in line |
| tx ntx ; | Move cursor to one char before the next occurance of letter "x" in line Move cursor to one char before the "n"th occurance of letter "x" in line. Go to one char before the next occurance in line |
| Tx nTx ; | Move cursor backwards to one char before the next occurance of letter "x" Move cursor backwards to one char before the "n"th occurance of letter "x" Go to one char before previous occurance in line |

Where search strings can have the following patterns:

|  |  |
| --- | --- |
| **Pattern** | **Description** |
| . | A period matches any single character |
| ^ | Finds the beginning of a line |
| ^A | Finds the beginning of a line where the first character is the letter 'A' |
| $ | Matches the end of a line |
| [abc] | Matches a string which contains any of the letters (a, b or c) between the brackets |
| \ | Turn off the special meaning of a character. Example "\." does not match the period to any character but to the period character specifically |
| \d | Match any single digit (0 to 9) |
| \* | A search expression followed by a '\*' matches zero or more of the search expression. For example "A\*" will match A, AA and AAA |
| + | Same as '\*' above except that it matches one or more of the search expression. |
| ? | Same as '\*' and "+" except that it matches zero or one occurances |
| string1|string2 | Match any either string 1 or string 2 |
| a.b | Matches a string beginning with the letter 'a' followed by any character, again followed by the letter 'c' |
| ^.$ | Matches an entire line containing only a single character |
| a(b\*|c\*)d | Matches a string beginning with the letter 'a' followed by zero or more of the letter 'b', followed by zero or more of the letter 'c' and then followed by the letter 'd' |
| Linux.\*Linux | Finds a line containing two instances of the string "Linux" |
| .\* [a-z]+ .\* | Finds a line containing a word comprised of all lower case letters with a single blank on either side of the word |

**Information Commands:**

|  |  |
| --- | --- |
| **Keystrokes** | **Action** |
| ctrl-g or :f | List file info: fine name, number of lines in file, position of cursor in file. |
| :set list :set nolist | Show tabs and end of line markers Turn of tab and eol markings |
| :args | Show command line arguments used |

Terminate session:

* Use command: ZZ  
  Save changes to current file and quit.
* Use command line: ":wq"  
  Save (write) changes to current file and quit.
* Use command line: ":w"  
  Save (write) changes to current file without quitting.
* Use command line: ":w!"  
  Save (write) changes to current file overriding the file permissions if the user has the privileges to change the file permissions. For example this will save a file with read only privileges if the user is the owner or has the ability to modify the privileges to allow a write. This will not permanently modify the file privileges. Note that there is no space between the two characters. A space will infer that the output is streamed to a Unix command following the "!".
* Use command line: ":w *filename*"  
  Save (write) changes to a new file of name "*filename*" without quitting.
* Use command line: ":q!"  
  Ignore changes and quit. No changes from last write will be saved.
* Use command line: ":qa"  
  Quit all files opened.

New session:

* Use command: ":e filename"  
  Start new edit session on specified file name without closing current vi / vim editor process.

|  |
| --- |
| **Vi/Vim modes:** |

Vi/Vim modes are set using the "set" command and its many options.

**:set all** - display all mode settings of your editing session.   
**:set termcap** - display terminal settings of your editing session.

**:set ic** - Change default to ignore case for text searches  
Default is changed from noignorecase to ignorecase. (ic is a short form otherwise type **set ignorecase**)

Common options to set:

|  |  |  |
| --- | --- | --- |
| **Full "set" Command** | **Short form** | **Description** |
| :set autoindent :set noautoindent | :set ai :set noai | {CR} returns to indent of previous line.  Turn on autoindent: :set ai  Turn off autoindent: :set noai  Set indent width: set shiftwidth=4  Intelligent auto-indent: set smartindent  Toggle autoindent on/off when pasting text (press F2 key to toggle mode after one is in "insert" mode): set pastetoggle=<F2> |
| :set autowrite :set noautowrite | :set aw :set noaw | This tells vim to automatically write the file when switching to edit another file. See tags, editing multiple files (next, rewind) |
| :set backspace=indent,eol,start :set backspace | :set bs=indent,eol,start :set bs | Allow backspacing over an indent, line break (end of line) or start of an insert |
| :set backup=on :set backup=off | :set bk=on :set bk=off | Create backup file of file changes while editing.  To automatically remove the backup file after the file being edited is written, use the option :set writebackup=on/off  File backup mode settings: :set backupcopy=yes/no/auto |
| :set cryptmethod=zip :set cryptmethod=blowfish |  | Set file encryption for file save of buffer contents.   * zip: pkzip * blowfish: strong encryption   This is set upon reading a file if encrypted. Vim 7.3+ |
| :set errorbells :set noerrorbells | :set eb :set noeb | Silence error beep |
| :set flash :set noflash | :set fl :set nofl | Screen flashes upon error (for deaf people or when noerrorbells is set) |
| :set tabstop=8 | :set ts | Tab key displays 8 spaces |
| :set ignorecase :set noignorecase | :set ic :set noic | Case sensitive searches |
| :set number :set nonumber | :set nu :set nonu | Display line numbers |
| :set showmatch :set noshowmatch | no abbreviations | Cursor shows matching ")" and "}" |
| :set showmode :set noshowmode | no abbreviations | Editor mode is displayed on bottom of screen |
| :set showmatch :set noshowmatch | no abbreviations | Cursor shows matching ")" and "}" |
| :set syntax on :set syntax off | no abbreviations | Set syntax highlighting and color highlighting for a file type (eg XML, HTML, C++, Java, etc). Also cursor shows matching ")" and "}" Also can set syntax highlighting explicitly: :set syntax=html Syntax definition files: /usr/share/vim/vim73/syntax/ |
| :set taglength | :set tl | Default=0. Set significant characters |
| :set closepunct='".,;)]} |  | % key shows matching symbol. Also see showmatch |
| :set linelimit=1048560 |  | Maximum file size to edit |
| :set wrapscan :set nowrapscan | :set ws :set nows | Breaks line if too long |
| :set wrapmargin=0 :set wrapmargin=8 :set nowrapmargin | :set wm   :set nowm | Define right margin for line wrapping. Wrap when past 8 characters from the edge of column display (often default 80). |
| :set list :set nolist |  | Display all Tabs and Ends of lines (Dislays these hidden characters). |
| :set bg=dark :set bg=light |  | VIM: choose color scheme for "dark" or "light" console background. |

See full list of [set options](http://vimdoc.sourceforge.net/htmldoc/options.html)

|  |
| --- |
| **Advanced "vi" features** |

**Interactive Command Examples:**

* **Marking a line:**

Any line can be "Book Marked" for a quick cursor return.

* + Type the letter "**m**" and any other letter to identify the line.
  + This "marked" line can be referenced by the keystroke sequence "**'**" and the identifying letter.  
    Example: "**mt**" will mark a line by the identifier "t".  
    "**'t"** will return the cursor to this line at any time.  
    A block of text may be referred to by its marked lines. i.e.**'t,'b**
  + Write a marked block to a new file: **'t,'bw *newfile***
* **vi line buffers:**

To capture lines into the buffer:

* + Single line: "**yy**" - yanks a single line (defined by current cursor position) into the buffer
  + Multiple lines: "**y't**" - yanks from current cursor position to the line marked "t"
  + Multiple lines: "**3yy**" - yank 3 lines. Current line and two lines below it.

Note all lines deleted are also stored in numbered line buffers.

Copy an entry in the buffer to the editing session:

* + "**p**" - place contents of latest entry to the buffer after current line defined by current cursor position. The buffer can be referenced by its number as well. The latest entry to the buffer is entry "1". Recall with the keystroke "1p
  + Each deleted line will end up in the vim line buffer. To recall the prior entry to the buffer use the keystroke: "2p. The prior entries to the buffer can all be referenced by its number. Each time an entry to the buffer is made, its position in the stack is incremented. Typically vim has nine (1-9) default "numbered" buffers.
* **vi named line buffers:**

Storage buffers can be named with letters of the alphabet: a-z.   
To capture lines into the buffer:

* + Single "yanked" line stored in buffer "a": **"ayy** - yanks a single line (defined by current cursor position) into the buffer named "a"
  + Deleted line stored in buffer "b": **"bdd** - deletes a single line (defined by current cursor position) into the buffer named "a"
  + Deletes 4 lines and stores in buffer "t": **"t4dd** - deletes four lines (defined by current cursor position) into the buffer named "a"

To paste lines from a named buffer:

* + Single line stored in buffer "a": **"ap** - paste contents of the buffer named "a" after the current line (defined by current cursor position)
* **vim: Shift a block of code left or right:**
  + Enter into visual mode by typing the letter "v" at the top (or bottom) of the block of text to be shifted.
  + Move the cursor to the bottom (or top) of the block of text using "j", "k" or the arrow keys.  
    Tip: Select from the first collumn of the top line and the last character of the line on the bottom line.  
    Zero ("0") will move the cursor to the first character of a line and "$" will move the cursor to the last character of the line.
  + Type >> to shift the block to the right.  
    Type << to shift the block to the left.

Note: The number of characters shifted is controlled by the "shift width" setting. i.e. 4: ":set sw=4"  
This can be placed in your $HOME/.vimrc file.

* **vim: Shift a block of code left or right (method #2):**
  + **:20,40>**   
    Shift text from row 20 to 30, to the right
  + **:20,40<**   
    Shift text from row 20 to 30, to the left

**Command Line:**

* **command options:**

The vi command line interface is available by typing "**:**". Terminate with a carriage return.  
Example commands:

* + **:help *topic***  
    If the exact name is unknown, TAB completion will cycle through the various options given the first few letters. Ctrl-d will print the complete list of possibilites.
* **Executing Unix commands in vi:**

Any UNIX command can be executed from the vi command line by typing an "!" before the UNIX command.  
Examples:

* + **:!pwd** - shows your current working directory.
  + **:!ls** - shows files in your current working directory.
  + **:sh** - open a new Bash shell. Editing session is suspended until you exit the shell. Execute all the commands you want and then return to the vim session.
* **Reading and merging/including external text:**
  + **:r *filename*** - include the contents of an external file
  + **:r !date** - reads the results from the date command into a new line following the cursor.
  + **:r !ls -1** - Place after the cursor, the current directory listing displayed as a single column.
* **Line numbers:**

Lines may be referenced by their line numbers. The last line in the file can be referenced by the "$" sign.  
The entire file may be referenced by the block "**1,$**" or "**%**"  
The current line is referred to as "**.**"  
A block of text may be referred to by its line numbers or its marked lines. i.e. **5,38** or **'t,'b**   
Write out a block of text denoted by line numbers **:5,38 w *newfile***   
Append a marked block to an existing file: **'t,'bw >> *filename***

* **Find/Replace:**

Example:

* + **:%s/fff/rrrrr/** - For all lines in a file, find string "fff" and replace with string "rrrrr" for the first instance on a line.
  + **:%s/fff/rrrrr/g** - For all lines in a file, find string "fff" and replace with string "rrrrr" for each instance on a line.
  + **:%s/fff/rrrrr/gc** - For all lines in a file, find string "fff" and replace with string "rrrrr" for each instance on a line. Ask for confirmation
  + **:%s/fff/rrrrr/gi** - For all lines in a file, find string "fff" and replace with string "rrrrr" for each instance on a line. Case insensitive.
  + **:'a,'bs/fff/rrrrr/gi** - For all lines between line marked "a" (ma) and line marked "b" (mb), find string "fff" and replace with string "rrrrr" for each instance on a line. Case insensitive.
  + **:5,20s/fff/rrrrr/gc** - For all lines between line 2 and line 20, find string "fff" and replace with string "rrrrr" for each instance on a line. Confirm each change with y/n.
  + **:1,$s/$/XXX/** - For all lines in the file, append a tripple X (XXX)
  + **:1,$s/XXX$//** - For all lines in the file, remove the tripple X (XXX)
  + **:%s/ \*$/** - For all lines in a file, delete blank spaces at end of line (there is a single space before the asterisk). Repeat with a tab instead of a space to delete trailing tabs.
  + **:%s/\(.\*\):\(.\*\)/\2:\1/g** - For all lines in a file, move last field delimited by ":" to the first field. Swap fields if only two.
  + **:%s#<[^>]\+>##g** - Find and remove all HTML tags but keep the text contents.
  + **:%s/^\(.\*\)\n\1$/\1/** - Find and remove all duplicate lines

For more info type:

* + **:help substitute**
  + **:help pattern**
  + **:help gdefault**
  + **:help cmdline-ranges**
* **Sorting:**

Example:

* + Mark a block of text at the top line and bottom line of the block of text. i.e. "**mt**" and "**mb**" on two separate lines. This text block is then referenced as **"'t,'b**.
  + Sort lines in block: (man page: [sort](http://man.yolinux.com/cgi-bin/man2html?cgi_command=sort))   
    **:'t,'b !sort**
  + Reverse order of lines in block: (man page: [tac](http://man.yolinux.com/cgi-bin/man2html?cgi_command=tac))   
    **:'t,'b !tac**
  + Sort lines of text in a paragraph. Block of lines defining the paragraph are identified by the cursor as the top and the first blank line as the end of the paragraph. Place curson on the line "Blue chair" and type the following:   
    **!}sort**   
    File to edit:

|  |
| --- |
| Blue chair  Red table  Green grass  Black stone  Other stuff goes here...  and here |

* + Becomes

|  |
| --- |
| Black stone  Blue chair  Green grass  Red table  Other stuff goes here...  and here |

* + Note that lines below the blank line delimeter are not sorted.
  + Sort lines of text in a paragraph by the second collumn:   
    **!}sort -f -k2**   
    option "-f" : ignore case   
    option "-k" : list collumn number to sort by   
    For a list of all options, see the [sort man page](http://man.yolinux.com/cgi-bin/man2html?cgi_command=sort)   
    File in previous example becomes:

|  |
| --- |
| Blue chair  Green grass  Black stone  Red table  Other stuff goes here...  and here |

* + Sort lines of text in a paragraph and arrange into four collumns:   
    **!}sort | pr -4t**   
    option -4 : four collumns   
    option -t : omit page headers and trailer   
    [pr man page](http://man.yolinux.com/cgi-bin/man2html?cgi_command=pr)   
    File to edit:

|  |
| --- |
| Blue  Red  Green  Black  Yellow  Orange  White  Brown  Other stuff goes here...  and here |

* + Becomes

|  |
| --- |
| Black Brown Orange White  Blue Green Red Yellow  Other stuff goes here...  and here |

* + Note that lines below the blank line delimeter are not sorted.
* **Moving columns, manipulating fields and awk:**

**:'t,. !awk '{print $3 " " $2 " " $1}'** - This will reverse the order of the columns in the block of text. The block of text is defined here as from the line marked with the keystroke **"bt"** and the current line (**"."**). This text block is referenced as "**'t,.**" (man page: [awk](http://man.yolinux.com/cgi-bin/man2html?cgi_command=awk))

aaa bbb ccc ccc bbb aaa

xxx yyy zzz becomes-> zzz yyy xxx

111 222 333 333 222 111

* **Source Code Formatting:** C++/Java
  + Use vim visual text selection to mark the lines to format (beautify):
    - eg. Whole file:
      * Go to first line in file: shift-v
      * Go to last line in file: shift-g
      * Select the key equals: =

This will align all braces and indentations. For the equivalent in emacs see the [YoLinux emacs tutorial](http://www.yolinux.com/TUTORIALS/LinuxTutorialXemacs.html).

* **Text Formatting:**
  + Mark a block of text at the top line and bottom line of the block. i.e. "**mt**" and "**mb**" on two separate lines.
  + Example: "**:'t,'b !nroff**"
  + You can insert nroff commands i.e.:

|  |  |
| --- | --- |
| .ce 3 | Center the next three lines |
| .fi | Fill text - left and right justify (default) |
| .nf | No Fill |
| .ls 2 | Double line spacing |
| .sp | Single line space |
| .sv 1.0i | Vertical space at top of page space |
| .ns | Turn off spacing mode |
| .rs | Restore spacing mode |
| .ll 6.0i | Line length = 6 inches |
| .in 1.0i | Indent one inch |
| .ti 1.0i | Temporarily one time only indent one inch |
| .pl 8.0i | Page length = 8 inches |
| .bp | Page break |

* + Example:

|  |
| --- |
| .fi  .pl 2i  .in 1.0i  .ll 6.0i  .ce  Title to be centered  .sp  The following text bla bla bla bla bla bla bla bla bla bla  bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla  bla bla bla bla bla bla bla bla bla bla bla bla bla bla  bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla  bla bla bla bla bla |

* + Becomes:

|  |
| --- |
| Title to be centered  The following text bla bla bla bla bla bla bla bla  bla bla bla bla bla bla bla bla bla bla bla bla  bla bla bla bla bla bla bla bla bla bla bla bla  bla bla bla bla bla bla bla bla bla bla bla bla  bla bla bla bla bla bla bla bla bla bla bla bla  bla bla bla bla |

* man pages:
  + [nroff](http://man.yolinux.com/cgi-bin/man2html?cgi_command=nroff) - text formatter (emulate nroff command with groff)
  + [troff](http://man.yolinux.com/cgi-bin/man2html?cgi_command=troff) - troff processor of the groff text formatting system
  + [tbl](http://man.yolinux.com/cgi-bin/man2html?cgi_command=tbl) - table formatter (for troff)
* **Text Width Formatting:**
  + Mark a block of text or reference a block by their line numbers and pipe them through fmt, a text formatter which splits lines on word boundaries.
  + **:20,30 !fmt -80** will re-format the lines from line 20 to line 30 to wrap at an 80 collumn margin. Any line longer than 80 characters (eg a long URL), will not get split. A line split occurs at word delimiters such as a blank space.

man page: [fmt](http://man.yolinux.com/cgi-bin/man2html?cgi_command=fmt)

* **Spell Checking:**
  + Mark a block of text by marking the top line and bottom line of the block. i.e. "**mt**" and "**mb**" on two separate lines.
  + **:'t,'b !spell** will cause the block to be replaced with misspelled words.
  + Press "**u**" to undo.
  + Proceed to correct words misspelled.

man page: [spell](http://man.yolinux.com/cgi-bin/man2html?cgi_command=spell)

* **Vim/Vi Macros:**

**:map letter commands\_strung\_together**  
**:map** - lists current key mappings  
Example - **:map g n cwNEW\_WORD{ctrl-v}{esc}i{ctrl-v}{CR}**  
This example would find the next search occurrence, change the word and insert a line feed after the word. The macro is invoked by typing the letter "g".

* + Control/Escape/Carriage control characters must be prefixed with ctrl-V.
  + Choose a letter which is not used or important. (i.e. a poor choice would be "i" as this is used for insert)
* **Double spacing:**
  + **:%s/$/{ctrl-V}{CR}/g**  
    This command applies an extra carriage return at the end of all lines
* **Strip blanks at end of line:**
  + **:%s/{TAB}\*$//**
* **Delete all lines beginning with or matching a pattern:**
  + **:1,$ /^#/d**   
    Delete all (first to last line: 1,$ or g) comments lines in file. Delete all lines beginning (^) with "#" (specify text pattern).
  + **:g/#/d**   
    Delete all lines (g) containing comments (comments follow "#") in file. Delete all lines containing "#".
  + **:g!/^#/d**   
    Delete all lines except (g! or v) comment lines beginning (^) with "#".
* **Strip DOS ctrl-M's:**
  + **:1,$ s/{ctrl-V}{ctrl-M}//**   
    Note: In order to enter a control character, one muust first enter ctrl-v. This is true throughout vi. For example, if searching for a control character (i.e. ctrl-m): /ctrl-v ctrl-M If generating a macro and you need to enter esc without exiting the vi command line the esc must be prefixed with a ctrl-v: ctrl-v esc.
* **Convert tabs to spaces:**
  + **:% !expand -t4**   
    convert tabs to four blank spaces for the whole file (%).
  + **:20,30 !expand -t4**   
    convert tabs to four blank spaces for lines 20 through 30.

Man page: [expand](http://man.yolinux.com/cgi-bin/man2html?cgi_command=expand)

* **Editing multiple files:**
  + **vi file1 file2 file3**
  + **:n** Edit next file (file2)
  + **:n** Edit next file (file3)
  + **:rewind** Rewind to the first file (file1)  
    or **shift-ctrl-~**
  + **:rewind!** Rewind to the first file (file1) without saving changes

Note: Named buffers from "yanked" and deleted lines are shared between files. Contents of unamed buffers are not. To save file changes when switching files :set autowrite

* **Line folding:**

Many times one may encounter a file with folded lines or may wish to fold lines. The following image is of a file with folded lines where each "+" represents a set of lines not viewed but a marker line prefixed with a "+" is shown stating how many lines have been folded and out of view. Folding helps manage large files which are more easily managed when text lines are grouped into "folds".

Example: vim /usr/share/vim/vim63/plugin/netrw.vim

**ssh-rsa AAAAB3NzaC1yc2EAAAABJQAAAQEAoNnhgvWknKRMUu2ATTK5nlzz01rCCTD3ypKbmYTHpFHcWm2ofe+wDsBG5q/JnunDV9/10WaBnaUlPSOlRQrKgAz2pTIaCxbJCnckM5wq27S+zWG5m0DZ8FUEPG6HWfSRfPZ2GpedN80FZDFHhOieDYkGaOqAwzJPuW38XdSv1S8CjTOy55T9uABqnfVpHoLxbk+X9GwCpjmIWnyHY/2PV2/mAaxXY7rLLktW7bTywNidUQDWyd1VA/4GcBu5GtuqVYyNCBL2j+2+Hjb1IY2lgrRmekSZobUsz0fP3nxRuQv8QOnvlhTEo62f7qKljU4dA7ntAYr2QmWyRgwNmgn/cw== rsa-key-20131105**

# How To Configure SSH Keys Authentication With PuTTY And Linux Server In 5 Quick Steps

Want to support HowtoForge? Become a [subscriber](http://www.howtoforge.com/subscription)!

|  |
| --- |
| Submitted by [EugenePik](http://www.howtoforge.com/forums/member.php?u=62329) ([Contact Author](http://www.howtoforge.com/forums/private.php?do=newpm&u=62329)) ([Forums](http://www.howtoforge.com/forums)) on Fri, 2009-06-26 15:38. :: [Linux](http://www.howtoforge.com/sitemap/linux) | [Security](http://www.howtoforge.com/sitemap/security)  How To Configure SSH Keys Authentication With PuTTY And Linux Server In 5 Quick Steps    This tutorial explains how you can replace password-based SSH authentication with key-based authentication which is more secure because only the people that own the key can log in. In this example, we're using [PuTTY](http://www.chiark.greenend.org.uk/%7Esgtatham/putty/) as our SSH client on a Windows system.   1. Get the zip file with all PuTTY binaries [http://the.earth.li/~sgtatham/putty/latest/x86/putty.zip](http://the.earth.li/%7Esgtatham/putty/latest/x86/putty.zip) 2. Generate a private and public key pair Open PuTTYgen.exe, press Generate button, move mouse. Once the keys are generated, type your key passphrase (choose a "hard to guess" one). Save Public key. Save Private key.  Screen shot: PuTTY key generator 3. Configure your Linux server (create user, save public key) For this guide let's assume you regular login name is autotimesheet (replace it with one that you use regularly).  As root, on the shell, type:  adduser autotimesheet --disabled-password  You will be asked to fill in some details such as the user's real name (empty string is fine).  Now type:  su autotimesheet cd /home/autotimesheet mkdir .ssh chmod 700 .ssh cd .ssh  Then in that folder, create and edit a file called authorized\_keys2. |

All future Raspbian images will ship with Oracle Java by default; existing users can install it by typing:

sudo apt-get update && sudo apt-get install oracle-java7-jdk

./Desktop/freedomotic-5.5.0.83-0504a4a/freedomotic-5.5.0.83-0504a4a/freedomotic.jar