WALES GENE PARK | PARC GENETEG CYMRU

Text editing on the UNIX command line

Beta version 1.1 (04 June 2013)

Contents

Nano

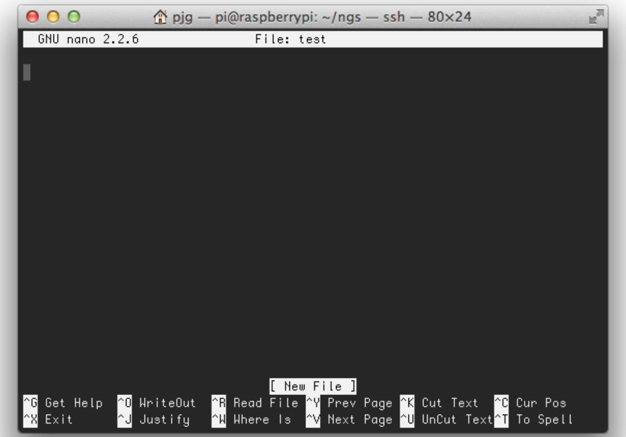
Nano is an easy to use UNIX text editor, which works from the command line without using a mouse or a complex graphical interface. It is similar to the "edit" program on old MS-DOS based computers.

The majority of commands in nano are initiated by holding down the control key and then pressing another character. So in this guide, ^x means to press and hold down the control key and then press x.

Starting nano

To edit a file called filename, type:

```
nano filename
```



```
pjg — pi@raspberrypi: ~/ngs — ssh — 80×24
  GNU nano 2.2.6                File: test




                              [ New File ]
^G Get Help  ^O WriteOut  ^R Read File ^Y Prev Page ^K Cut Text  ^C Cur Pos
^X Exit      ^J Justify   ^W Where Is  ^V Next Page ^U UnCut Text^T To Spell
```

Most of the important commands are listed at the bottom of your screen.

Editing text

Insert new text at the current cursor position just by typing in the text.

Navigation

Use the arrow keys to move around the page in nano.

Other navigation commands

`^a`	Move to the beginning of a line
`^e`	Move to the end of a line
`^y`	Move down a page
`^v`	Move up a page
`^_`	Move to a specific line (^_^v moves to the top of the file, ^_^y to the bottom)
`^c`	Find out what line the cursor is currently on

Searching

```
^w        Search for some text
```

Having pressed ^w, you will then be prompted to enter the text you wish to search for. The search starts from the current cursor position and then searches down the file, wrapping back to the top of the file if the term isn't found before the end of the file.

Deleting text

```
^d        Delete character under the cursor
```

```
Backspace  Delete character in front of the cursor
```

```
^k        Delete entire line
```

```
^\        Search for (and replace) a string of characters
```

Cut and paste

The ^k command does not delete lines permanently. The most recent set of deletions are stored in a buffer that you can then recover. This may be a re-insertion at the same point using ^u (undo), or you can move the cursor elsewhere and recover the text (cut and paste).

```
^u        Recover the delete line buffer
```

To delete a block of text, you can just repeatedly use ^k.

To insert multiple copies of your text buffer you can just press ^u as many times as required.

Repeatedly use ^K until all of the text you want to move has been deleted. Move to the line that you want to insert the text at, and use ^U.

Saving and Exiting

```
^o        Save contents without exiting (you will be prompted
          for a file to save to)
```

```
^x        Exit nano (you will be prompted to save your file if
          you haven't)
```

```
^t        When saving a file, opens a browser that allows you
          to select a file name from a list of files and
          directories
```

In nano, you can insert another file:

```
^r        read an existing file into nano (inserted at the
          current cursor position)
```

Getting help

```
^G        nano help
```

vi

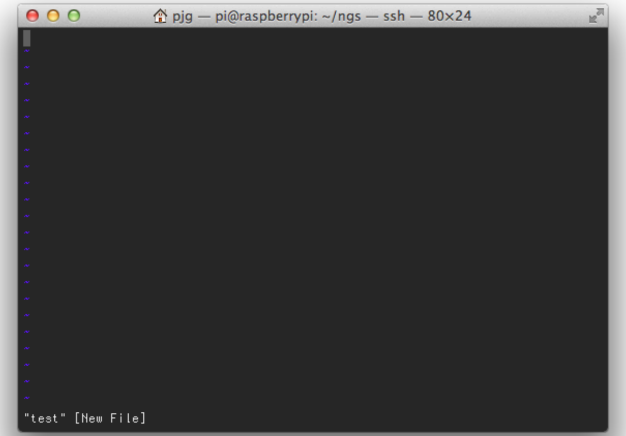
The default editor on all UNIX systems is called vi (visual editor). Novice users often find it tricky to use, because just like most UNIX commands there is very little in the way of visual clues as to what you are doing.

Many people question why you should learn vi, but its presence in the UINX standards and its presence on any UNIX system you come across, make learning the basics a very worthwhile use of your time.

Starting vi

To edit a file called filename, type:

```
vi filename
```



```
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
"test" [New File]
```

Command and insert modes

There are two key modes that vi operates in, command mode and insert mode. Insert mode is entered using certain keystrokes (e.g. a), and can be exited (back into command mode) by pressing the escape key.

`<esc>`	Exits insert mode

Inserting or adding text

Each of the following commands put the vi editor into its insert mode. Therefore, you need to press escape to exit this mode back into command mode before you can switch into a different mode.

`i`	Insert text before cursor, until <Esc> hit
`I`	Insert text at beginning of current line, until <Esc> hit
`a`	Append text after cursor, until <Esc> hit
`A`	Append text to end of current line, until <Esc> hit
`o`	Open and put text in a new line below current line, until <Esc> hit
`O`	Open and put text in a new line above current line, until <Esc> hit

Editing text

The following commands allow you to modify text.

```
r           Replace a single character under cursor
            (no <Esc> needed)
```

```
R           Replace characters, starting with current cursor
            position, until <Esc> hit
```

```
cw          Change the current word with new text, starting with
            the character under cursor, until <Esc> hit
```

```
C           Change (replace) the characters in the current line,
            until <Esc> hit
```

```
cc          Change (replace) the entire current line, stopping
            when <Esc> is hit
```

Navigation

```
j or <Return> or <down-arrow> Move cursor down one line
```

```
k or <up-arrow> Move cursor up one line
```

```
h or <Backspace>  or <left-arrow> Move cursor left one
character
```

```
l or <Space> or <right-arrow>

          Move cursor right one character
```

```
0 (zero)  Move cursor to start of current line (the one with
          the cursor)
```

```
:n<Return> or nG    Move cursor to line n
```

```
:$<Return> or G     Move cursor to last line in file
```

Screen manipulation

^f	Move forward one screen
^b	Move backward one screen
^d	Move down (forward) one half screen
^u	Move up (back) one half screen

Searching

/string	Search forward for occurrence of string in text
?string	Search backward for occurrence of string in text
n	Move to next occurrence of search string
N	Move to next occurrence of search string in opposite direction

Deleting text

x	Delete single character under cursor
dd	Delete entire current line

Undo

u	Undo last action

Cut and paste

```
yy         Copy (yank, cut) the current line into the buffer
```

```
p          Put (paste) the line(s) in the buffer into the text
           after the current line
```

Saving and reading files

```
:r filename<Return> Read file named filename and insert after
           current line
```

```
:w<Return> Write current contents to file named in original vi
           call
```

```
:w newfile<Return>  Write current contents to a new file named
           newfile
```

```
:w! prevfile<Return>     Write current contents over a pre-
           existing file named prevfile
```

Exiting

```
:x<Return>     Quit vi, writing out modified file to file
          named in original invocation
```

```
:wq<Return>    Quit vi, writing out modified file to file
          named in original invocation
```

```
:q<Return>     Quit (or exit) vi
```

```
:q!<Return>    Quit vi even though latest changes have not
          been saved for this vi call
```