**Using vim buffers to copy lines**

The vim editor has a total of 27 buffers: 26 named buffers (a–z) and 1 unnamed buffer that is overwritten by each new operation.

Unless you have specified a named buffer, all operations use the unnamed buffer, and two operations right after each other have the second operation overwrite the data from the first operation.

Named buffer operations are always preceded by the double quotation mark (“), which tells the machine that a named buffer operation is to follow.

When you perform named buffer operations, the buffers must be referred to in the command as either a lowercase or uppercase letter of the alphabet (which is part of the command and not sent to the buffer):

**Lowercase buffer letter—**Overwrites the buffer

**Uppercase buffer letter—**Appends to buffer

For example, the following string, when entered in Command mode with the cursor in column 1, causes line 1 to copy the next three lines to the named buffer (a), overwriting any contents of that named buffer:

"a3yy

The vim editor allows you to copy text from your file into temporary buffers for use in other places in the file during your current vim work session. Each buffer acts like temporary memory, or, as some programs call it, a "clipboard" where you can temporarily store information. The simplest case would be using the vim command mode command:

**Press Escape to return to editor mode (normal mode, command mode), then input**

**:u**

**You can use :u to redo several times**

The **u** (undo) command undoes the most recent thing you've done in vi. If you've deleted something by accident, the **u** command will restore it to your file. Whenever you delete something from a file, vim keeps a copy in a temporary file called the **general buffer**. You can also delete or copy lines into temporary files called named buffers that will let you reuse those lines during your current vim work session. This feature is particularly useful if you are editing a file that contains a lot of boilerplate text or repetitive code.

When you quit vi, all the buffers are emptied; they are temporary storage that only lasts for your current vim work session.

Here is how to cut-and-paste or copy-and-paste text using a visual selection in Vim.

**Cut and paste:**

1. Position the cursor where you want to begin cutting.
2. Press **v** to select characters (or uppercase **V** to select whole lines). (or you can use cursor to select multiple lines)
3. Move the cursor to the end of what you want to cut.
4. Press **d** to cut (or **y** to copy).
5. Move to where you would like to paste.
6. Press **P** to paste before the cursor, or **p** to paste after.

**Copy and paste** is performed with the same steps except for step 4 where you would press **y** instead of **d**:

* **d** = delete = cut
* **y** = yank = copy

## Contents

[[show](http://vim.wikia.com/wiki/Copy,_cut_and_paste)]

## Multiple copying[Edit](http://vim.wikia.com/wiki/Copy,_cut_and_paste?action=edit&section=1)

Deleted or copied text is placed in the unnamed register. If wanted, a register can be specified so the text is also copied to the named register. A register is a location in Vim's memory identified with a single letter. A double quote character is used to specify that the next letter typed is the name of a register.

For example, you could select the text hello then type "ay to copy "hello" to the a register. Then you could select the text world and type "by to copy "world" to the b register. After moving the cursor to another location, the text could be pasted: type "ap to paste "hello" or "bp to paste "world". These commands paste the text after the cursor. Alternatively, type "aP or "bP to paste before the cursor.

### Windows clipboard[Edit](http://vim.wikia.com/wiki/Copy,_cut_and_paste?action=edit&section=2)

When using Vim under Windows, the clipboard can be accessed with the following:

* In step 4, press Shift+Delete to cut or Ctrl+Insert to copy.
* In step 6, press Shift+Insert to paste.

### Different instances[Edit](http://vim.wikia.com/wiki/Copy,_cut_and_paste?action=edit&section=3)

How to copy and paste between two instances of Vim on different Linux consoles?

After copying text, open a new buffer for a new file:

:e ~/dummy

* Paste the text to the new buffer.
* Write the new buffer (:w).
* Switch to the previous buffer (:bp) to release \*.swp.
* Now switch to the other console.
* Put the cursor at the desired place.
* Read the dummy file (:r ~/dummy)

## Increasing the buffer size[Edit](http://vim.wikia.com/wiki/Copy,_cut_and_paste?action=edit&section=4)

By default, only the first 50 lines in a register are saved, and a register is not saved if it contains more than 10 kilobytes. [:help 'viminfo'](http://vimdoc.sourceforge.net/cgi-bin/help?tag=%27viminfo%27)

In the example below, the first line displays the current settings, while the second line sets:

* '100 [Marks](http://vim.wikia.com/wiki/Using_marks) will be remembered for the last 100 edited files.
* <100 Limits the number of lines saved for each register to 100 lines; if a register contains more than 100 lines, only the first 100 lines are saved.
* s20 Limits the maximum size of each item to 20 kilobytes; if a register contains more than 20 kilobytes, the register is not saved.
* h Disables [search highlighting](http://vim.wikia.com/wiki/Highlight_all_search_pattern_matches) when Vim starts.

:set viminfo?

:set viminfo='100,<100,s20,h

In the vi editor, how can commands such as moving the cursor or copying liens into the buffer be issues multiple times or applied to multiple rows:

By specifying the number right in front of a command such as 4l or 2yj