



## Using Emacs with BixChange

Unless you are using an editor that works in different environments, you might want to learn to use Emacs. Emacs has been ported to a wide variety of platforms so you are certain to always have a familiar environment to work in.

Emacs is superior to standard editors for several reasons. The most notable include the ability to have multiple buffers open simultaneously and the ability to add customizations. With Emacs Lisp, you can make the editor do wonders.

## This Document's Conventions

Code or command prompt sections will look like this:

```
fp> echo 'hello world' > foo.txt
```

Emacs uses a variety of key sequences to get things done. With a key sequence, you typically press and hold one key and have to press a few others in succession. The letter 'C' represents the Control key and 'M' denotes the Alt key.

An example of a sequence is C-f, which means you press and hold (the dash means 'hold') the Control key and press the 'f' key on the keyboard. This sequence C-x C-s means to hold the Control key and press and hold both the 'x' and 's' keys all as one sequence. If the sequence was C-f s, then you would release the Control and 'f' key and then subsequently press the 's' key all by itself.

## Prerequisites

You will need to install Emacs for your platform. If you are using Linux you most likely already have Emacs or can install it by accessing your setup. For Windows

you will need to download the binary at

<http://www.gnu.org/software/emacs/windows/faq2.html>

If you haven't already, you will need to install BixChange ([www.bixchange.com](http://www.bixchange.com)) to follow along with this tutorial.

## Opening and Closing Emacs

Open Emacs by clicking on the link in Windows or calling calling Emacs on the command line.



Figure 1: A Windows version of Emacs

Figure 1 is split into two windows. The top has the actual Latex file used to create this document. The lower window is a command line shell. You can do the same in your Emacs editor (and finally close it) by doing the following:

C-x 2	Opens second screen buffer
M-x shell	Opens a command shell
C-x C-c	Closes Emacs

Note that Emacs will ask you if you want to close the active shell session prior to closing Emacs. Actually type the word “yes” to the right of the question.

## Opening Buffers

Reopen Emacs and split the buffers again as in the previous section. When Emacs first opens up, unless you have done some customizations, you will always wind up with a buffer called `*scratch*`. This buffer is great if you are trying out some Lisp (the programming language Emacs was built in) programming or do some basic mathematical calculations. Otherwise, it is just a place to dump scratch information. The contents of this buffer are not saved, so keep important things out of it.

When you want to create a new file use `C-x C-f` and type in the path in the small window at the bottom. Windows Emacs is smart enough to use long path names as well as slashes the face the wrong way (Windows likes the backslash) as in:

```
Find file: c:/temp/wonderMan.txt
```

This presumes that the directory exists. You will run into trouble if you try and create buffers in directories that do not exist, so make sure your directory is a real one.

By the way, all of these shortcut key sequences map to commands with long names. The Find file you see above is actually a command in Emacs called, you guessed it, `find-file`. Try the following:

```
M-x find-file
```

When you hit enter the exact same line appears.

## Buffers are Just That

A buffer is not the real file, just an in-memory representation. The real file is safe on disk until you decide to update it with the contents of the buffer. Emacs is good about automatically backing up your buffer though. If you look in the directory where your original file exists you will notice another file with the same name but

with a squiggly line at the end of the name. That is the Emacs backup file. If something dire goes wrong with your file, you can usually use that backup file.

## **Killing Buffers**

Type something interesting in your buffer. In order to save the buffer, you use C-x C-s. The status window at the bottom of your Emacs window will tell you whether your file actually saved. In order to kill your current buffer you can use M-x kill-this-buffer.

## **Finding Text and Escaping**

When you have a buffer open, use C-s to find text. Type in the search term in the bottom window.

After Emacs finds the first instance of your search term it will lead you to it and highlight it. If you want to find other instances of the same word just use C-s again without retyping anything. Emacs will continue to find other instances of the word or phrase each time you repeat the sequence.

If you find yourself in a command you do not like, hit the Esc (escape) key three times. This sequence will normally get you out of whatever mode you are in.

## **Beginning and End Of Line and Document**

You can get to the beginning of a line by using C-a. The end of the line can be quickly reached by using C-e. There are key sequences for moving up and down a single line or left and right, but the directional keys on your keyboard are probably easier to use and remember.

Going to the beginning and end of the document can be achieved in two ways, depending on how your Emacs is configured. If one does not work, try the other: M-Shift-< or C-Home will get you to the top of the page. M-Shift-> or C-End will get you to the end.

## **Replacing Text and Reverting the Buffer**

If you find that you want to replace some text use M-x replace-string. First type in the string you want to replace and next the term or phrase to replace it with. When complete, Emacs will tell you how many replacements were made. If you

do not want to make these changes permanent, just use `M-x revert-buffer`. This will dump the changes you have made and reload the file on disk into the buffer window.

## Read Only Buffers

You open a normal read/write buffer with `C-x C-f`, but you open a read-only one with `C-x C-r` or by using `M-x find-file-read-only` and pressing Enter.

## Switching Between Buffers

Switching between visible buffers is as easy as using `C-x o`. Try bouncing back and forth between your buffers. You will need this as some terminal sessions do not recognize your mouse so this will be the only way to get around between the buffers.

If you want to see a buffer that is loaded but that you cannot currently see, use `C-x b` and type in the name of the buffer.

## Fixing Mistakes

If you make a mistake the easiest thing to do is to `M-x undo`. Another option is to use `M-x revert-buffer`.

## Loading Files

The commands you have been using up to this point have been created in Lisp and stored in files with the `.el` extension. BixChange currently has only one such file to assist you in performing an important operation – loading up BixChange modules.

So-called BixChange modules are literally zipped (actually tarred) files that have a `.dat` file in them with all of the components of the module. When you are creating a website, you will follow this concept, but more on that later.

The `bxit.el` file is in the `/bin` directory under the BixChange install. After you have it open, you will need to change the `this-dir` variable, shown here:

```
(setq this-dir "c:/WebServ/wwwroot/cgi-bin/bixchange")
```

The `setq` command in Emacs Lisp sets a variable in the file. In order to use `bxit.el` you will need to change the directory in quotes to the one where you installed BixChange.

## Using `bxit.el`

After you have made your modification of `bxit.el` either close it (M-x `close-this-buffer`) or reopen it in read-only mode (C-x C-r). This will prevent you from accidentally changing it.

In order to use the commands in `bxit.el` use M-x `load-file` and either type in the location of the file or use the up arrow to find it in the history. You should see the word *done* at the bottom of the screen.

The current version of `bxit.el` only has three commands. You can use their key sequences or simply type M-x and then the command name. They are:

C-c C-b n	bxit-new-ini
C-c C-b r	load-these-buffers
C-c C-b g	go-here

The key sequences *are* a bit long, so you can use M-x and the command in the right column to use any of these.

The first, `bxit-new-ini`, will create a new INI file. Do not type in the file extension. It will be added automatically.

Next, the `load-these-buffers` command expects the name of a `.dat` file (again, do not use the file extension) that should be in the same location as the `this-dir` variable that you modified. If you download any of the modules at <http://download.forwardphase.com> and you unpack it in the BixChange home directory, there will be a `.dat` file there you can experiment with.

Finally the `go-here` command basically emulates a feature Emacs already contains – the ability to jump to a buffer on a list. If your cursor is in the `.dat` file's buffer (it will be read-only) just use M-x `go-here` and hit Enter when you see OK at the bottom of the screen.

## **Finding Files From Strings**

Often you will look at something you have published and freak out – you spelled something wrong or a certain piece of information is not what you wanted to say. One of the easiest methods to find your information quickly is to gain access to your server via telnet or secure telnet (SSH) client, go to the BixChange install directory and use the grep command: `grep -ir “your search term” ./data`.

If you have split your Emacs windows, you can turn one of them into a shell (M-x shell) and use the same command there.

## **The Future**

This primer only scratches the surface of Emacs’ capabilities. If you want to learn more, try to Google ([www.google.com](http://www.google.com)) and hunt around for “Emacs tutorials”.