

UNIX Editors:

- **nano**
- **vi/vim**
- **emacs**



I believe that it to your advantage to learn to use a native text based editor on Unix/Linux.

I will present a bit of information about 3 Linux editors:

- nano
- vi
- emacs

What it comes down to it, I really don't care what editor you use. I care about the code you write.

GNU nano

- GNU nano is a plain text editor for Unix-like computing systems or operating environments using a **command line interface**.
- Nano emulates the Pico text editor, part of the Pine email client, and provides additional functionality.
- GNU nano puts a two-line "shortcut bar" at the bottom of the screen, listing many of the commands available in the current context.



```
GNU nano 2.3.1      File: my_cat1.c

#include <stdio.h>

#ifdef MAX_LINE_LEN
# define MAX_LINE_LEN 1024
#endif // MAX_LINE_LEN

int main(int argc, char *argv[])
{
    char line[MAX_LINE_LEN];
    char *line_ptr;

    while ((line_ptr = fgets(line, MAX_LINE_LEN, stdin)) != NULL) {
        fputs(line, stdout);
    }
}
```

[Read 17 lines]

^G Get Help	^O WriteOut	^R Read Fil	^Y Prev Pag	^K Cut Text	^C Cur Pos
^X Exit	^J Justify	^W Where Is	^V Next Pag	^U UnCut Te	^T To Spell

GNU nano

If you want colored code, as shown in the previous image, you can make a copy of my `.nanorc` file (from my home directory).

```
include /usr/share/nano/c.nanorc
```

```
include /usr/share/nano/makefile.nanorc
```

That's the end of the presentation on nano.

Though it is very basic, it does not require a lot from you.

There are files for other languages also available.

The Original Flame War



https://en.wikipedia.org/wiki/Editor_war

vi/vim

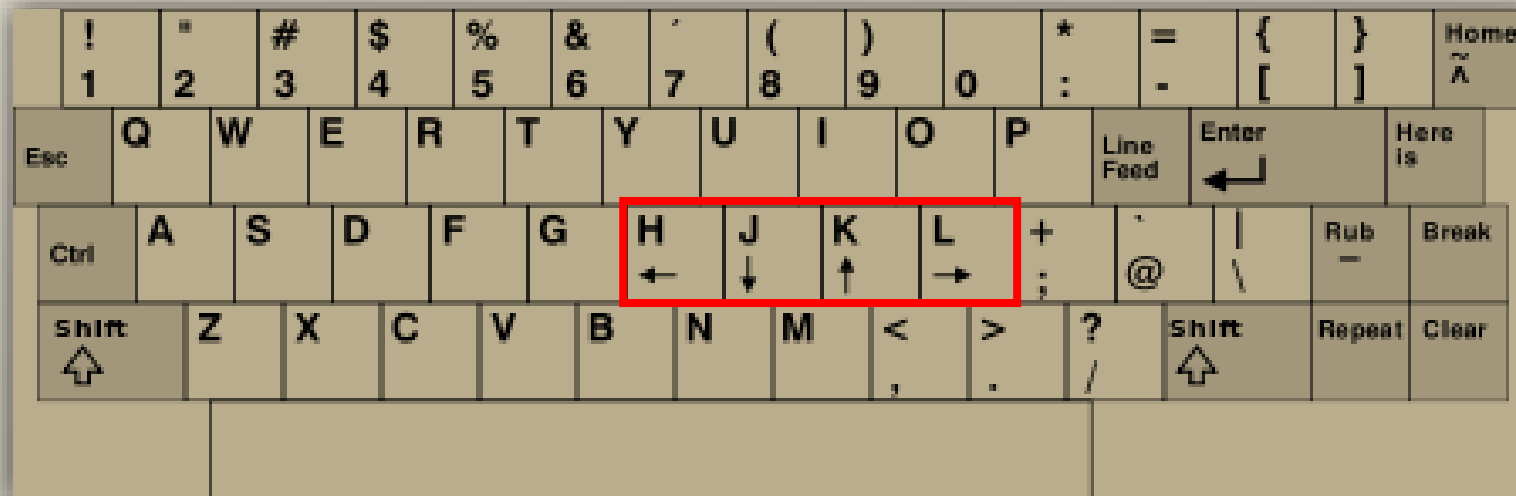
- "vi" is derived from the shortest unambiguous abbreviation for the `ex` command `visual` (`ex` is another UNIX editor)
- `vim` a contraction of Vi IMproved
- You will find `vi` in all UNIX/Linux implementations.
- Small and fast.

emacs

- Editing **MACroS**
- Non-modal interface.
- Extensible and customizable Lisp programming language variant.

- Regarding vi's modal nature, some emacs users *joke* that vi has two modes – **beep repeatedly and break everything**.
- vi users enjoy joking that emacs' key-sequences induce carpal tunnel syndrome, or mentioning one of many satirical expansions of the acronym EMACS, such as **Escape Meta Alt Control Shift** (a jab at emacs' reliance on modifier keys).
- As a poke at emacs' extensive programmability, vi advocates have been known to describe Emacs as "a great operating system, lacking only a decent editor".

- The original code for `vi` was written by Bill Joy in 1976, as the visual mode for a line editor called `ex` that Joy had written with Chuck Haley.
- `vi` was derived from a sequence of UNIX command line editors, starting with `ed`, which was a line editor designed to work well on teletypes, rather than display terminals.
- Joy developed the `vi` code on a ADM-3A terminal, which lacked many of the keys we now find common.



Bill Joy's words about vi and emacs:

*I think as mode-based editors go, it's (v*i*) pretty good.*

One of the good things about EMACS, though, is its programmability and the modelessness. Those are two ideas which never occurred to me.

<https://en.wikipedia.org/wiki/Vi>

More Joy (why vi doesn't have multiple windows, as emacs has):

What actually happened was that I was in the process of adding multiwindows to vi when we installed our VAX, which would have been in December of '78. **We didn't have any backups and the tape drive broke.** I continued to work even without being able to do backups. And then **the source code got crunched and I didn't have a complete listing.** I had **almost rewritten** all of the display code for windows, and that was when I gave up. After that, I went back to the previous version and just documented the code, finished the manual and closed it off. If that scrunch had not happened, vi would have multiple windows, and I might have put in some programmability – but I don't know.

Starting `vi`

```
$ vi [filename]
```

```
$ vim [filename]
```

- If `filename` does not exist, a screen will appear with just a cursor at the top followed by tildes (~) in the first column.
- If `filename` does exist, the first few line of the file will appear.
- The status line at the bottom of your screen shows error messages and provides information and feedback, including the name of the file.



tildes

This is what an empty file looks like in vi.

"JUNK" [New File]

0,0-1

All

```
#include <stdio.h>

#ifndef MAX_LINE_LEN
# define MAX_LINE_LEN 1024
#endif // MAX_LINE_LEN

int main(int argc, char *argv[])
{
    char line[MAX_LINE_LEN];
    char *line_ptr;

    while ((line_ptr = fgets(line, MAX_LINE_LEN, stdin)) != NULL)
    {
        fputs(line, stdout);
    }
}
```

"my_cat1.c" 17L, 284C 1,1 Top

vi Modes

Command Mode

- Command mode is the mode you are in when you start (default mode)
- Command mode is the mode in which commands are given to move around in the file, to make changes, and to leave the file
- Commands are case sensitive: j not the same as J
- Most commands do not appear on the screen as you type them. Some commands will appear on the last line: : / ?

vi Modes

Insert (or Text) Mode

- The mode in which text is created.
- You must press <Return> at the end of each line unless you've set wrap margin.
- There is more than one way to get into insert mode but only one way to leave: return to command mode by pressing the ESCAPE key, <Esc>

When in doubt about which mode you are in, start pounding on the <Esc> key.

Entering, Deleting, and Changing Text

From Command Mode

- i Enter text entry mode
- x Delete a character
- dd Delete a line
- r Replace a character
- R Overwrite text, press <esc> to end

I need a hat like this!



Eric was so proficient with the Unix text editor, he became known as the *vi*-king...

Exiting vi

To exit from vi, you must be in command mode

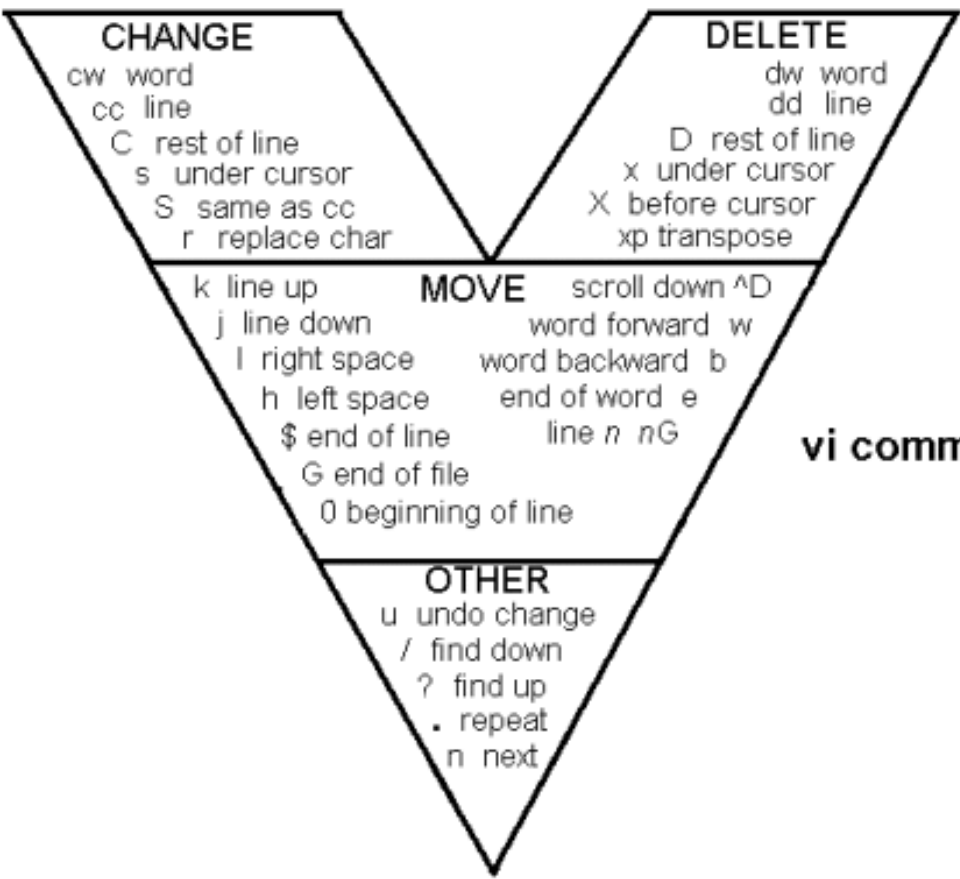
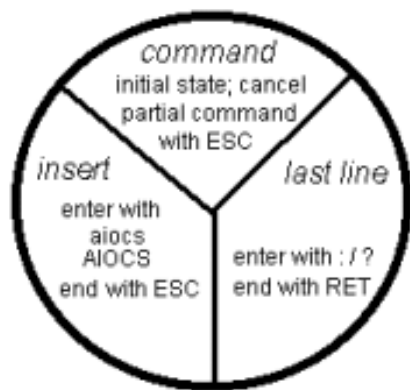
Press <Esc> if you are not in command mode.

You must press <Return> after commands that begin with a : (colon)

From Command Mode

- ZZ Write (if there were changes), then quit.
- :wq Write, then quit.
- :q Quit (will only work if file has not been changed).
- **:q! Quit without saving changes to file.**

vi states



vi commands

INSERT

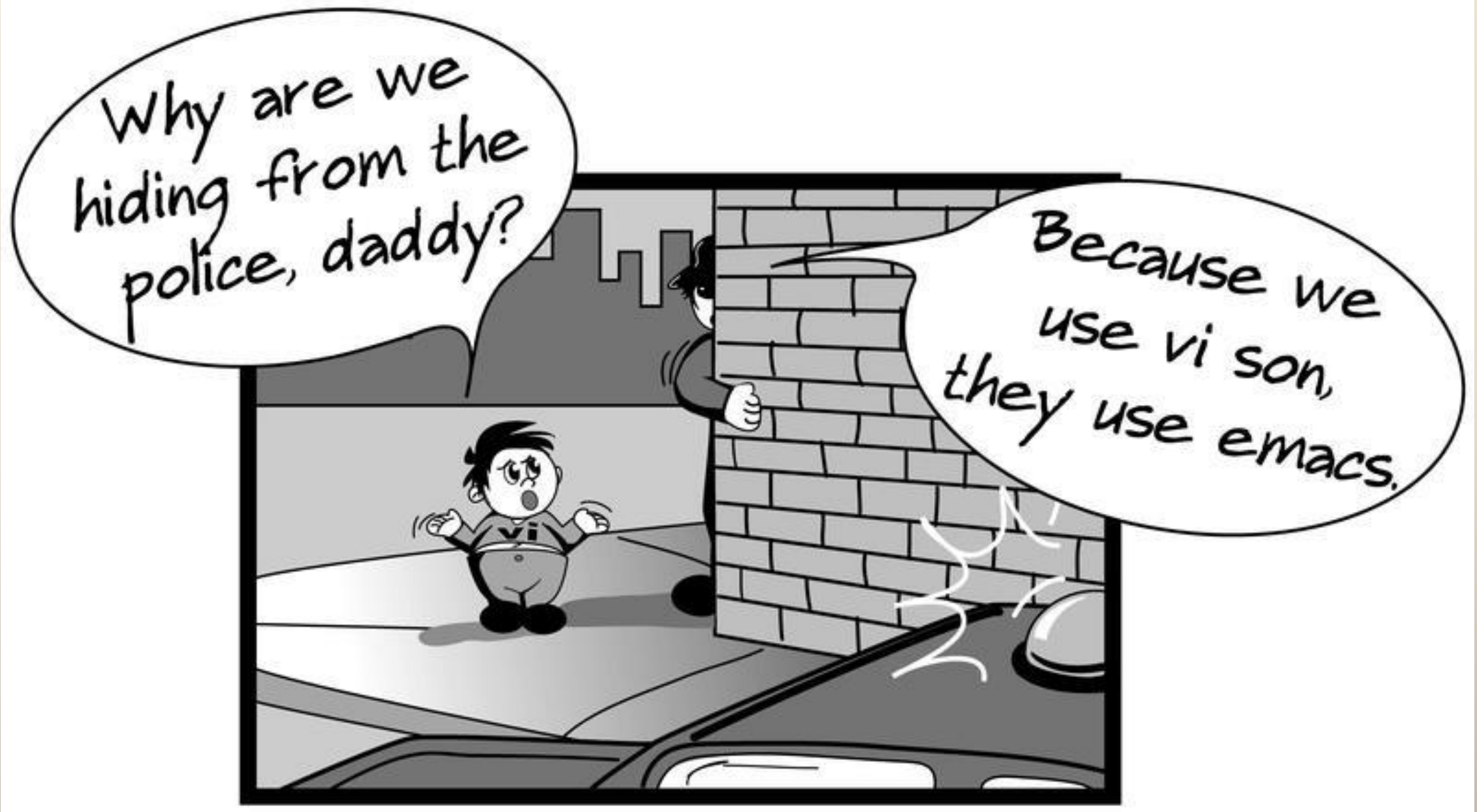
a	after cursor	yank word	yw
A	at end of line	yank line	yy
i	before cursor	put	p
I	at beginning of line		
o	open line below		
O	open line above		

ex COMMANDS

: se nu	set numbers
: se nonu	no numbers
:r file	read in file
:l cmd	run cmd
:se wm=10	wrap words

SAVE/QUIT

:w	write buffer
:q	quit
:wq	write & quit
:ql	abandon buffer
ZZ	same as :wq
^Z	suspend vi



- `emacs` began at the Artificial Intelligence Laboratory at MIT.
 - In 1972, staff hacker Carl Mikkelsen added display-editing capability to TECO, by using a set of macros written in the TECO macro language.
- The collection of macros was organized by Richard Stallman and was called TecoEmacs (for Editor MACroS)
 - In 1984, Stallman rewrote `emacs` in C, creating GNU `emacs` and (later) the collection of GNU software.

emacs Saves the World

It can be *reasonably* asserted that the writing of GNU `emacs` and the creation of FSF that followed, **started the revolution** of Open Source software that we enjoy so much now.

The **first** example of free and open-source software is believed to be the A-2 system, developed at the UNIVAC division of Remington Rand in 1953, which was released to customers with its source code.



emacs – Terminal Editor or IDE?

Yes.

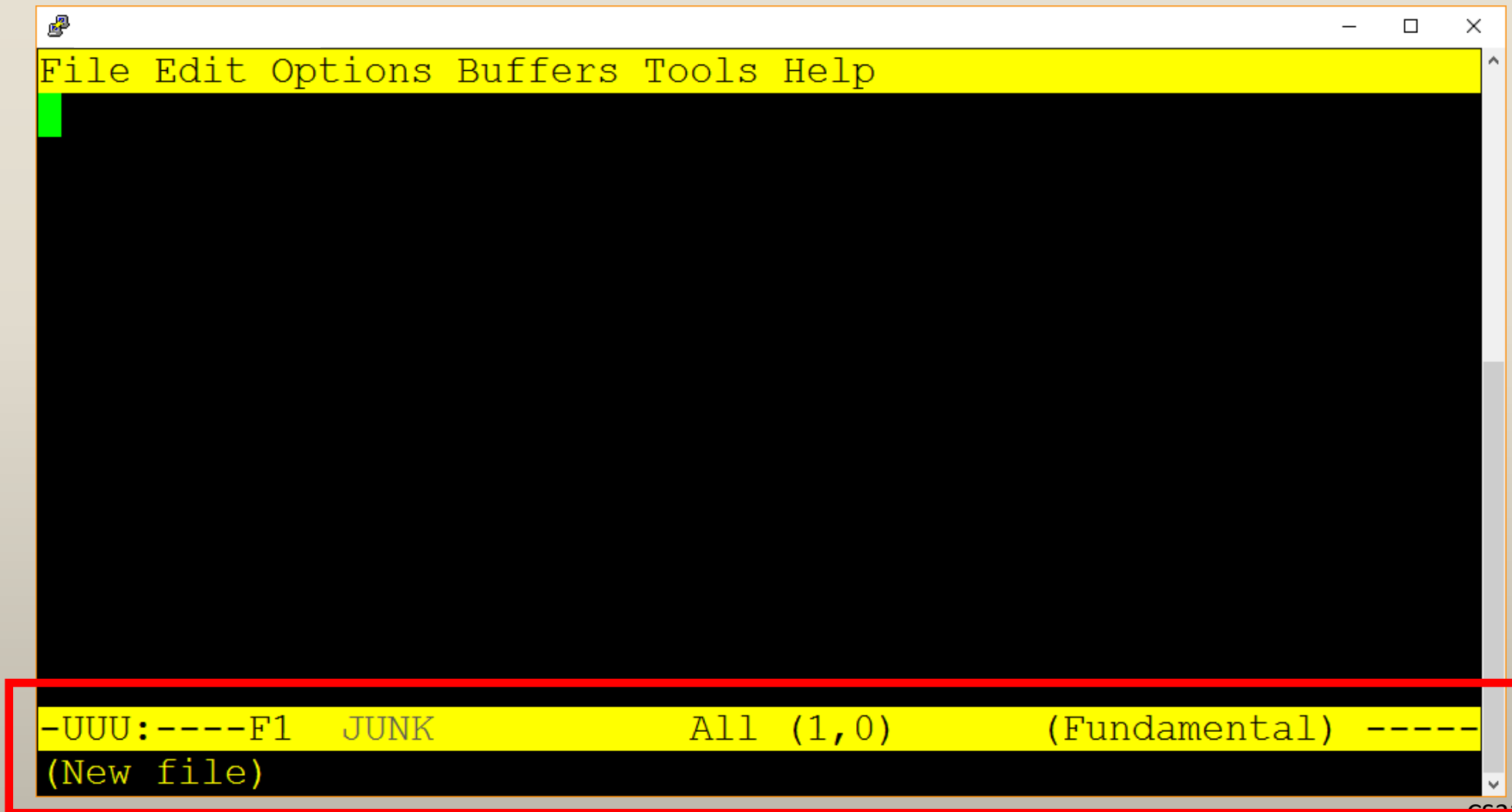
emacs can be used as strictly terminal based editor or as an IDE.

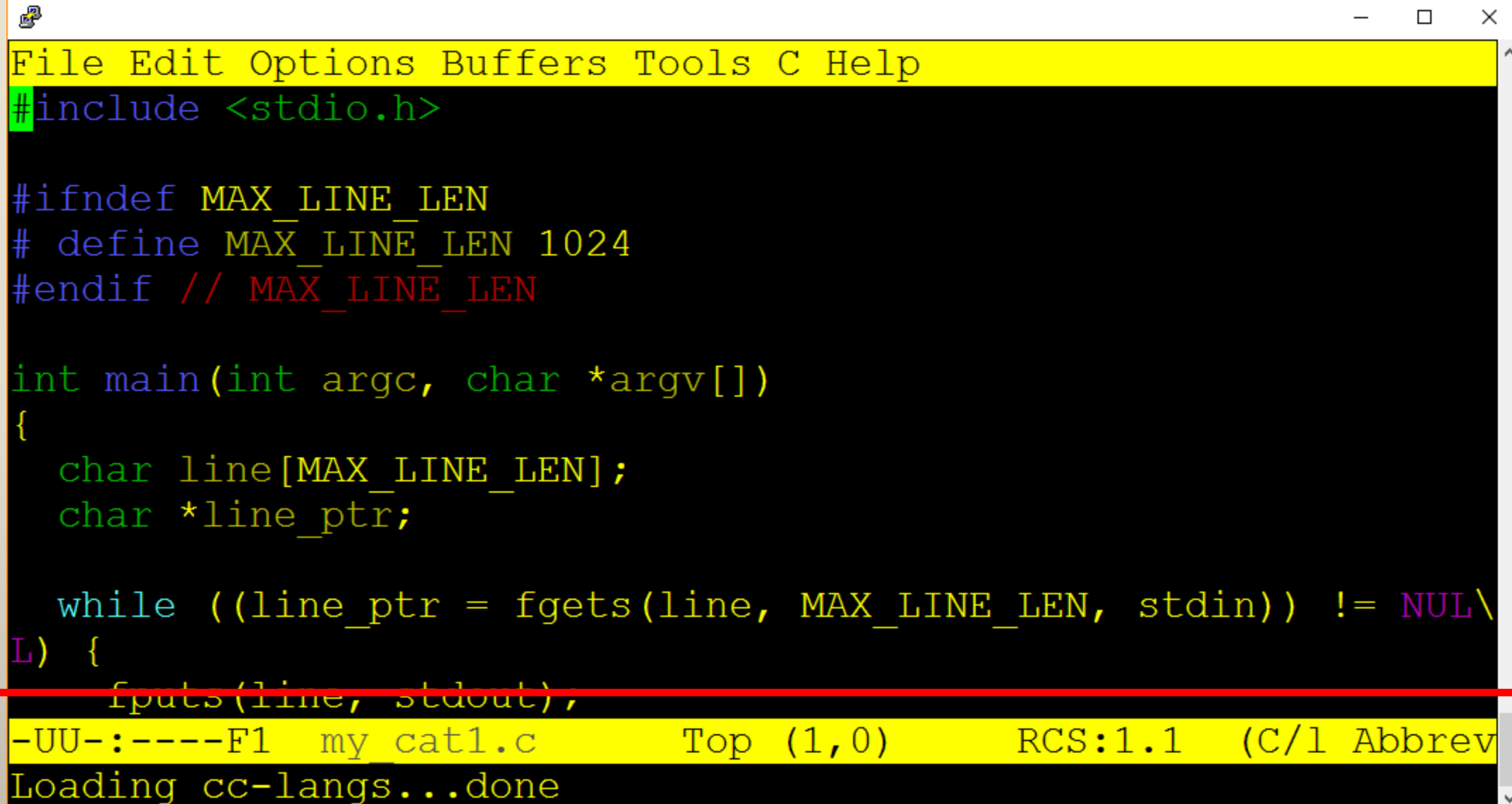
Compared to an IDE like Visual Studio, emacs may seem lacking. However, it is much more customizable than Visual Studio.

It is also free.

Starting emacs

```
$ emacs [filename]
```





```
File Edit Options Buffers Tools C Help
#include <stdio.h>

#ifndef MAX_LINE_LEN
# define MAX_LINE_LEN 1024
#endif // MAX_LINE_LEN

int main(int argc, char *argv[])
{
    char line[MAX_LINE_LEN];
    char *line_ptr;

    while ((line_ptr = fgets(line, MAX_LINE_LEN, stdin)) != NULL) {
        fputs(line, stdout),
- UU-:----F1  my cat1.c      Top (1,0)      RCS:1.1  (C/l Abbrev
Loading cc-langs...done
```

Where `vi` is modal (mo-dull) in nature, `emacs` is not.

- Rather `emacs` relies on **key-cords** and/or **key-sequences** for most/many commands.
- An example simple key-cord is the command that moves to the end of the current line:
 - **C-e** which means pressing the control key and the e key at the same time.
- An example of a key sequence is how to exit `emacs` (assuming there are no modified files)
 - **C-x C-c** which means pressing the control key with the x key, **releasing both**, then pressing the control key with the c key.

A key-cord in emacs is a combination of one or more of the modifier keys pressed with another key on the keyboard.

The modifier keys are:

- Control (abbreviation is C)
- Alt (abbreviation is A)
- Meta (abbreviation is M)
- Shift (abbreviation is S)

The meta key was common on the Lisp machines in use when emacs was developed at MIT.

On a Windows keyboard, **you can use the Alt key as the Meta key.**

You can also simulate the meta key by pressing (and releasing) the escape key as a key-sequence.

Key-sequence	Command
C-x C-c	Exit emacs – if there are unsaved files, you will need to confirm. This stands for Control-x Control-c
C-x C-s	Save current file.
C-x s	Save all files.
C-x C-f	Open file.
C-space	Set mark.
C-w	Kill region
C-y	Yank region back from kill.
C- / or C-x u	Undo previous edit.

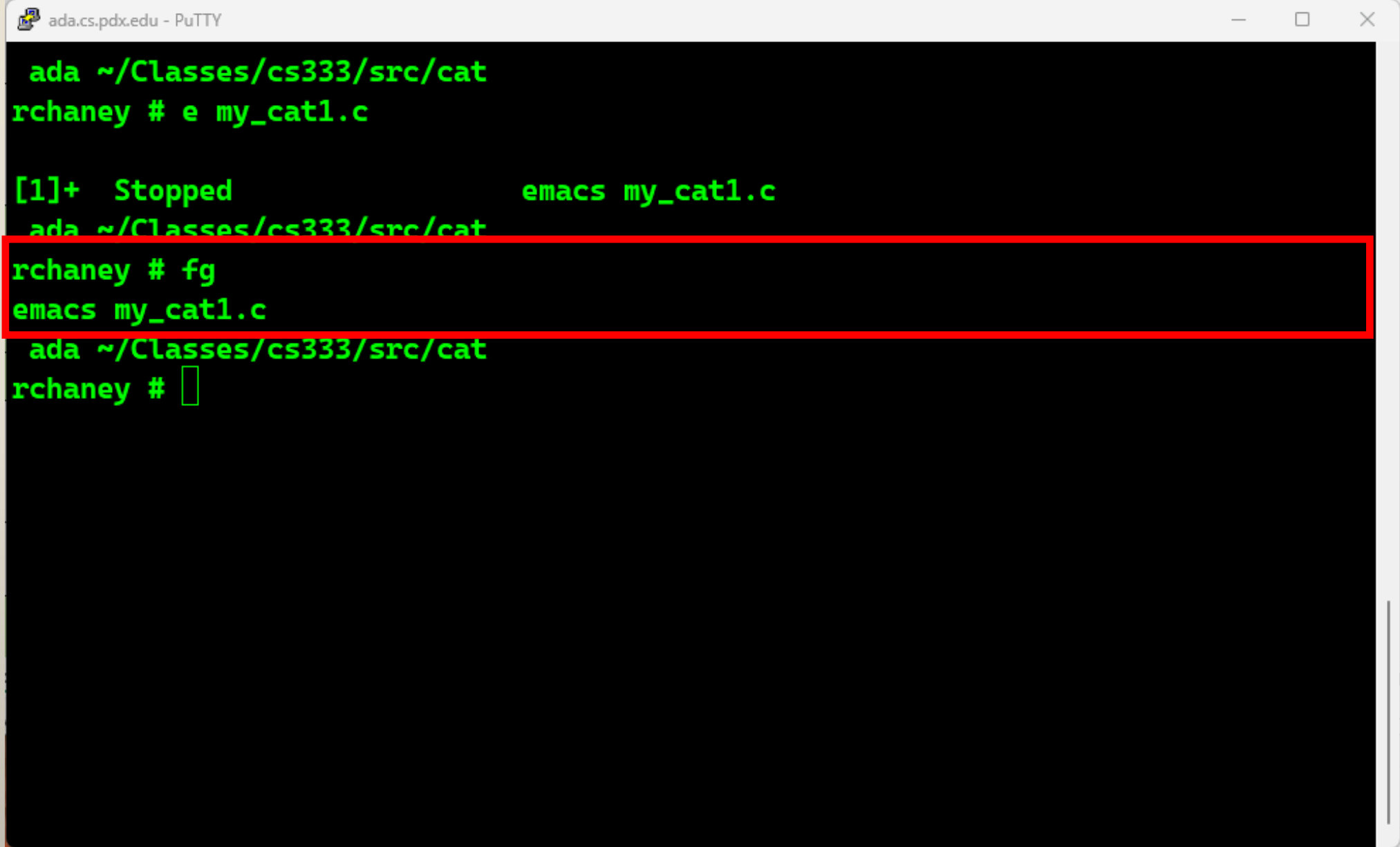
Key-sequence	Command
M- <code><</code>	Beginning of file. This is Meta-GreaterThan
M- <code>></code>	End of file
M-g g	Go to line
M-w	Kill region without delete (aka copy)
C-a	Beginning of line
C-e	End of line
C-l	Center window at point
C-x 1	Close other windows
C-x 2	Split current window horizontally

A Background Note

- You will very likely try to use the Windows version of undo (the control-z key-cord) to undo some command in either vi or emacs (or both).
- **The control-z key-cord in UNIX is very different from the control-z in Windows.**
- In a Windows application, control-z is probably the undo command.
- In UNIX, a control-z means to “push the current foreground application into the background”.
- If you press control-z and find yourself suddenly at the shell prompt, **don't rerun the editor**, return to your editor session by typing ‘`fg`’ (which stands for foreground) at the shell prompt.

```
ada.cs.pdx.edu - PuTTY
ada ~/Classes/cs333/src/cat
rchaney # e my_cat1.c

[1]+  Stopped                  emacs my_cat1.c
ada ~/Classes/cs333/src/cat
rchaney #
```



```
ada.cs.pdx.edu - PuTTY
ada ~/Classes/cs333/src/cat
rchaney # e my_cat1.c

[1]+  Stopped                  emacs my_cat1.c
ada ~/Classes/cs333/src/cat
rchaney # fg
emacs my_cat1.c
ada ~/Classes/cs333/src/cat
rchaney #
```