Programming Tools
Lecture 4

Lecturer: Liana Hovhannisyan

Command Line Editors

- Command line editors are exactly what they sound like. They give you *the ability to edit files from the command line*.
- There's a whole bunch of them, too:
  - Pico
  - Nano
  - Emacs
  - Vi
  - Vim
  - Neovim

Command Line Editors

- Nano (which is basically a clone of Pico) is probably the most common one. It's just a dead simple editor and most people can usually figure out how to use it by just opening it up.
- Vim on the other hand requires training. Vim is a clone of Vi but improved (**Vi IMproved**). It has all the functionality of Vi and more - things like extra features, plugins, and more.
- Vim is also extremely extensible. You can use it as your primary editor or just as a simple editor for changing files when SSH'd into a server.
- **Learning Vim is an investment.** As you learn it, you'll only get better with it and find more and more things to improve your productivity. Very good people with it will claim it's like an **extension of your fingers** allowing you to edit files faster and smarter.

VIM Editor

- Vim works in almost any OS environment - including Windows. You can expect to be able to use it on virtually any machine or system that you're working with.

Windows

For Windows users visit the Official Vim website (https://vim.sourceforge.io/download.php) to download.

Linux

Vim ships as a package for *nix systems.

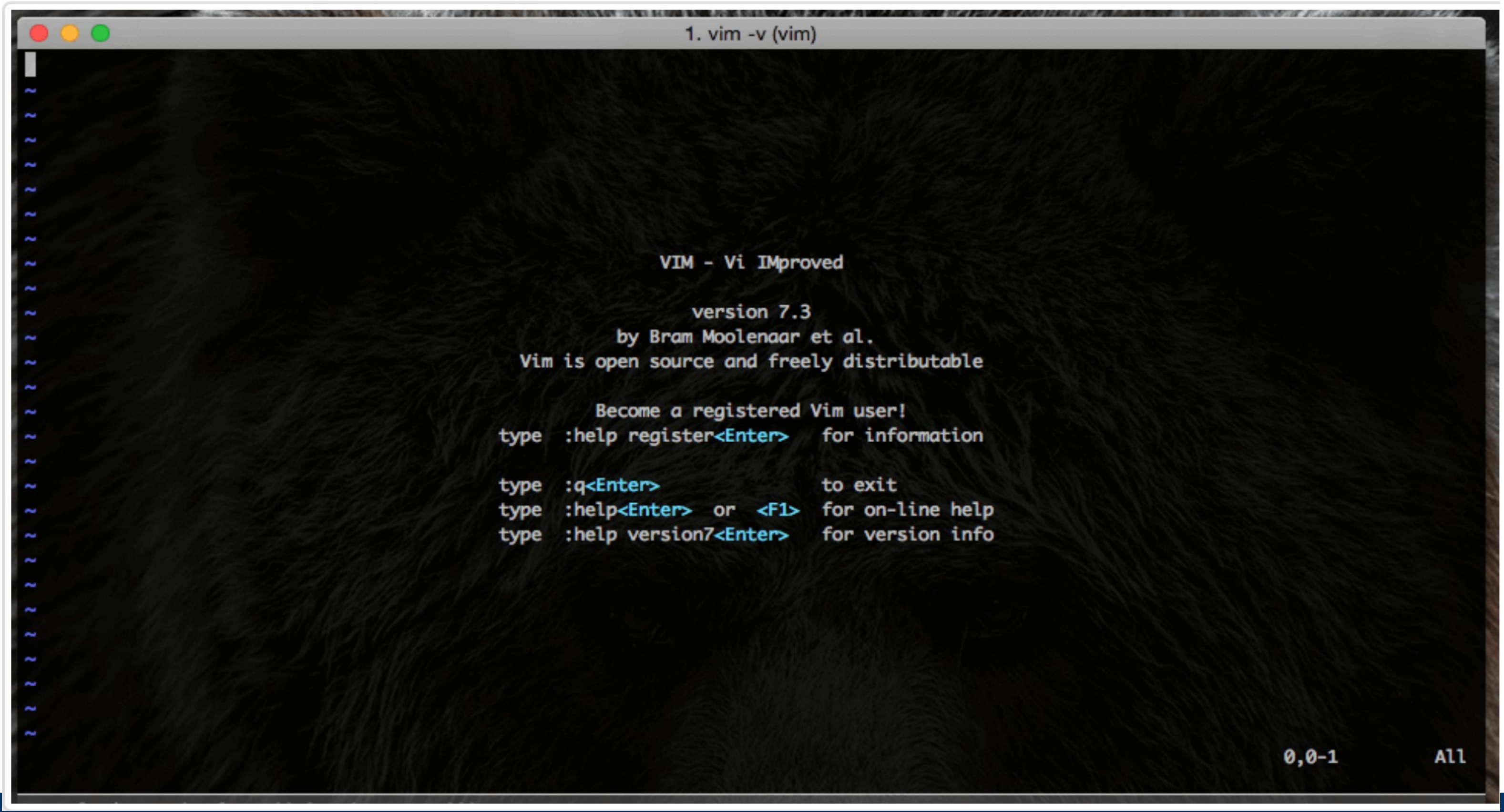
For Ubuntu, just run this from your terminal:

```
sudo apt-get install vim
```

VIM Editor

- Test your install. Now that you have installed (or updated) Vim, it's time to test to see if it worked. From the command line in your terminal, type:

  $ vim



```
1. vim -v (vim)

                         VIM - Vi IMproved

                           version 7.3
                     by Bram Moolenaar et al.
            Vim is open source and freely distributable

                    Become a registered Vim user!
          type  :help register<Enter>   for information

          type  :q<Enter>               to exit
          type  :help<Enter>  or  <F1>  for on-line help
          type  :help version7<Enter>   for version info

                                                        0,0-1         All
```

VIM Editor – Basic Starter

- That's it! Now to exit this screen, just type:

```
:q
```

- Before we go into detail, let's do a super basic starter example to get things rolling.

- From the terminal, navigate to a file and let's edit it with Vim:

```
$ vim whatever.txt
```

If the file exists we will edit it, otherwise we will create new file.

VIM - Modes

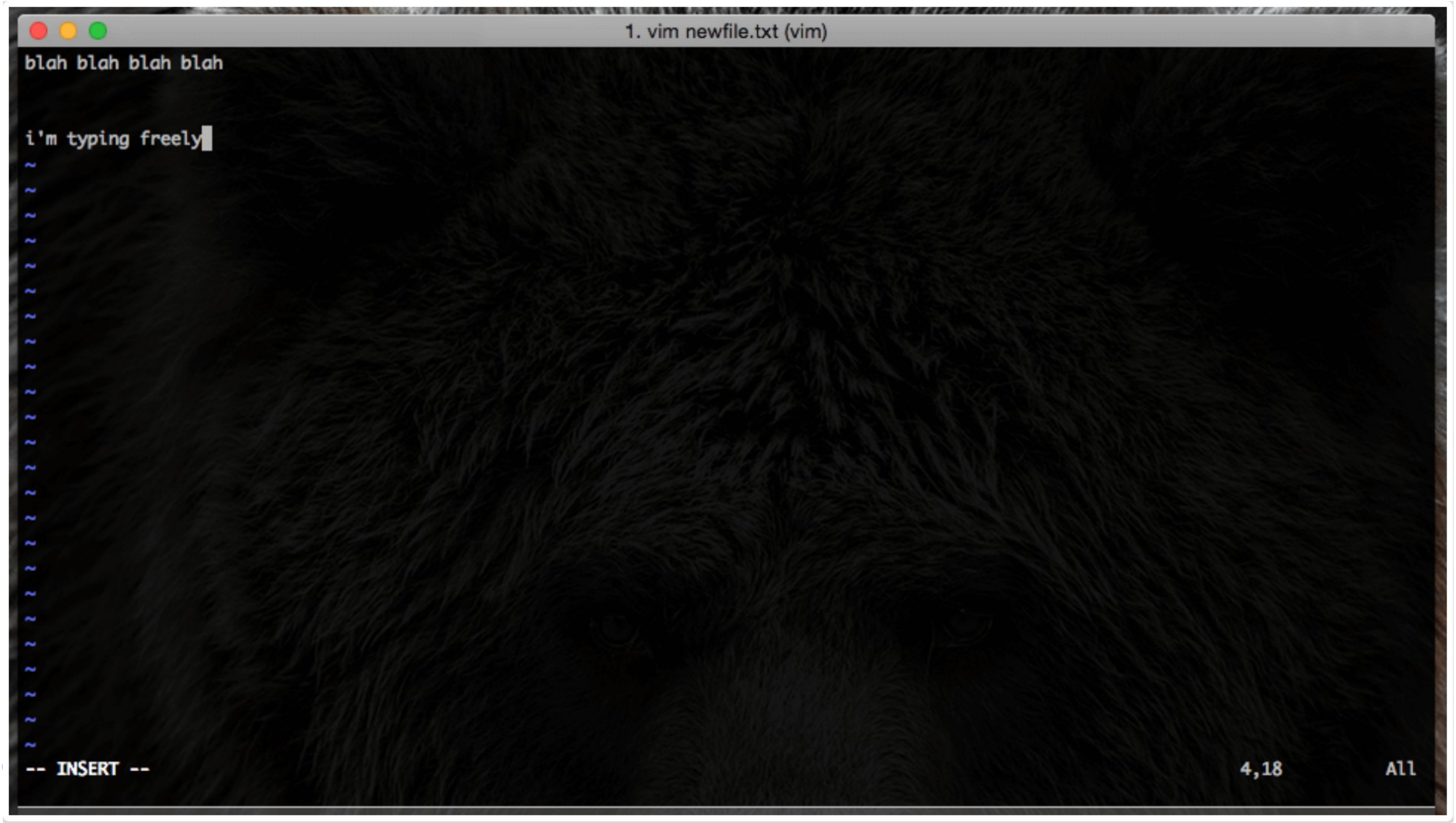
- Now that you're using Vim, we need to explain the two modes that Vim has:
  - **Command Mode**
  - **Insert Mode**.
- **Command Mode**, just like it sounds, is for executing commands. Things like custom Vim commands (we'll cover later), saving files, etc.
- **Insert Mode**, also just like it sounds, is for editing text freely.

VIM – Insert Mode

- To enter **Insert Mode** simply type:

  i

  and start typing (inserting) text.



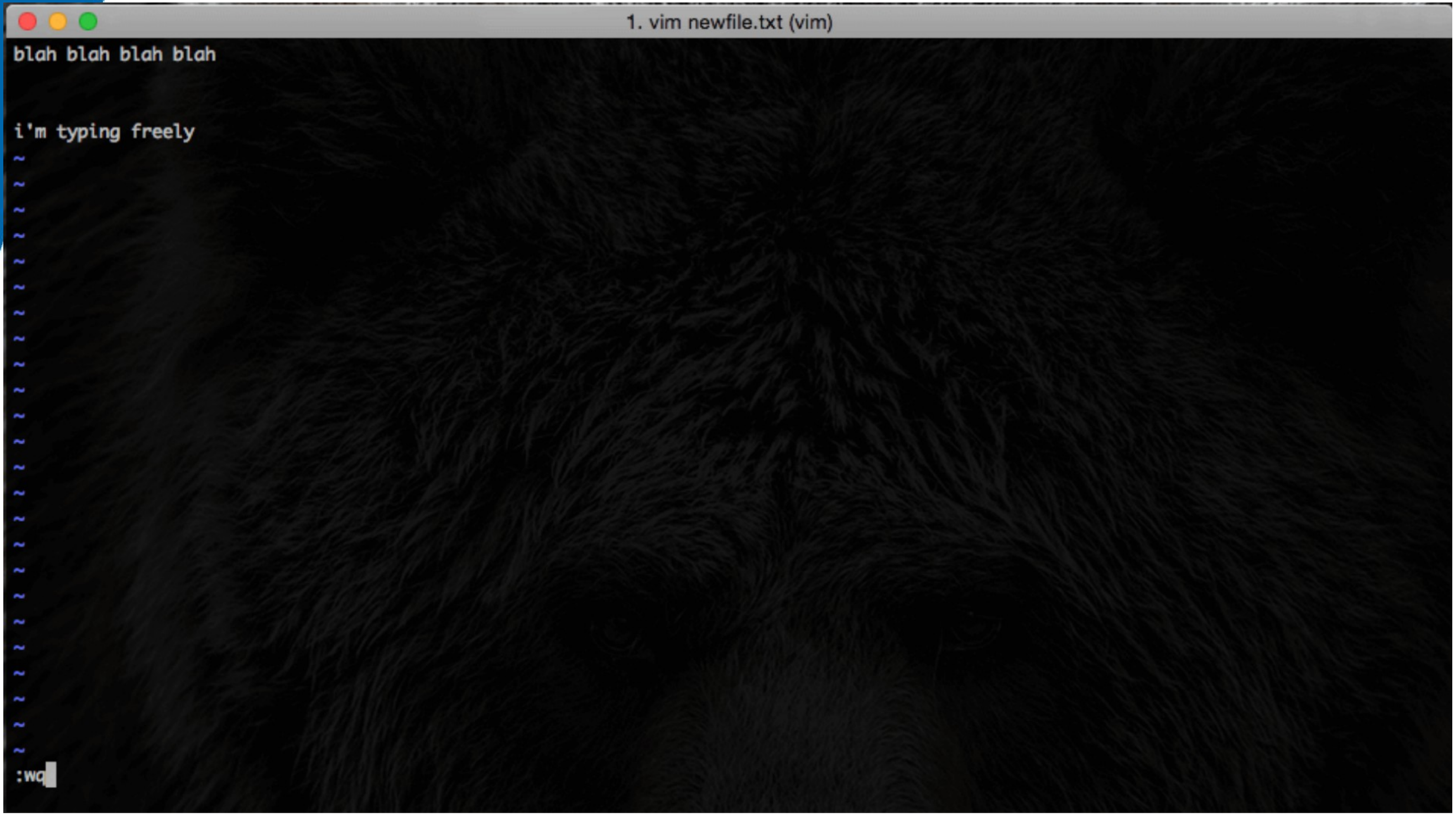
```
blah blah blah blah


i'm typing freely
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
-- INSERT --                                                          4,18          All
```

VIM – Command Mode

To save the file after editing you need to first exit **Insert Mode** and enter **Command Mode** by hitting ESC.

Once you're back into **command mode**, you'll need to save the file (called a Write) and then quit Vim. To enter a command, you need to hit the semicolon key :. Command to save the edits (**w**rite, **q**uit) is **:wq**

VIM – Command Mode



```
blah blah blah blah


i'm typing freely
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
:wq
```

VIM – Command Mode

Alternatively, if you want to quit Vim without saving changes, just type:

```
:q!
```

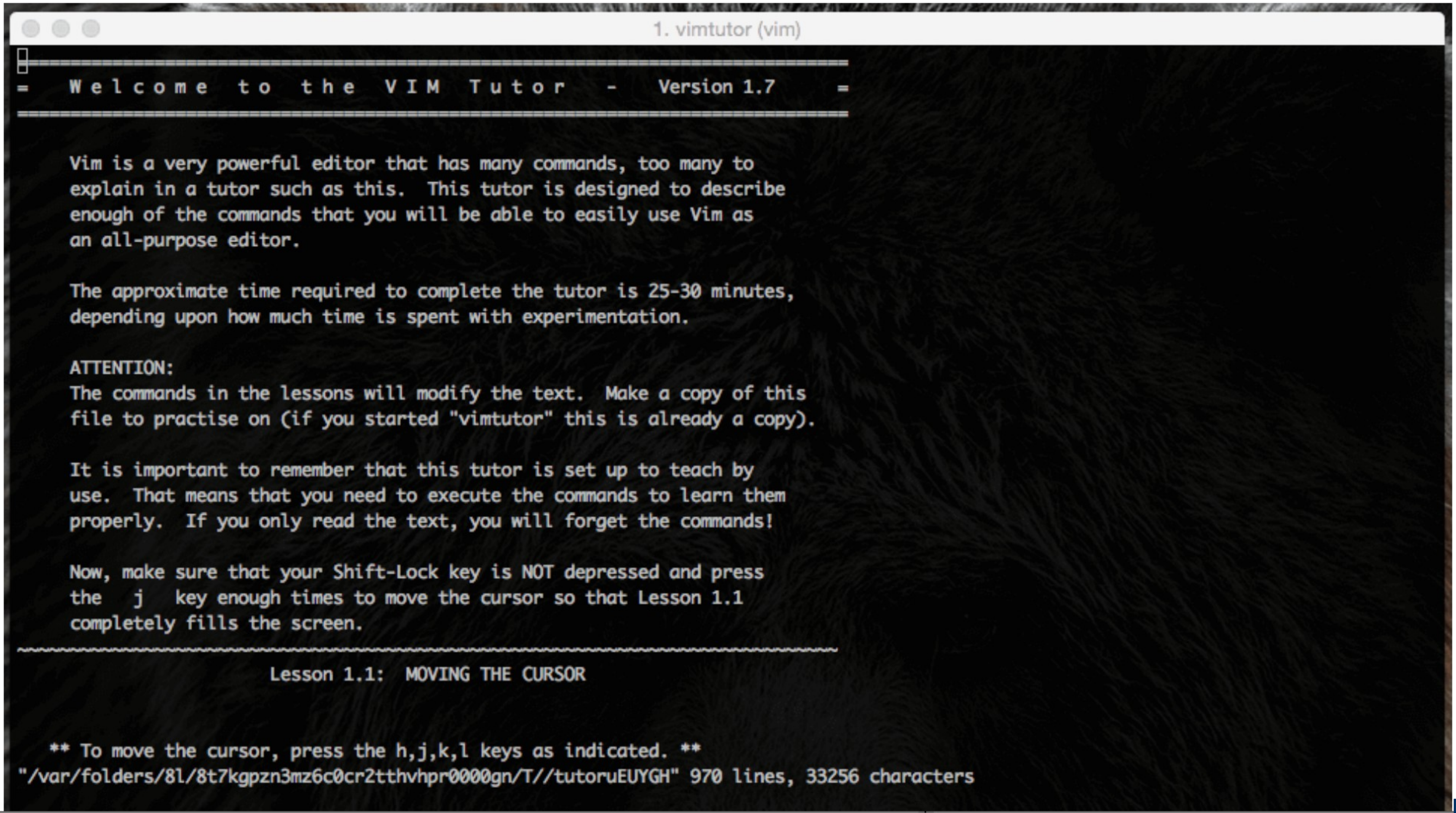
The exclamation mark means discard changes. So this literally will translate to "**q**uit and discard changes" for Vim.

VIM Tutorial

- **Vim also comes with its own tutorial**. If you need to freshen up on your skills, you can simply type this from the command line to bring it up:

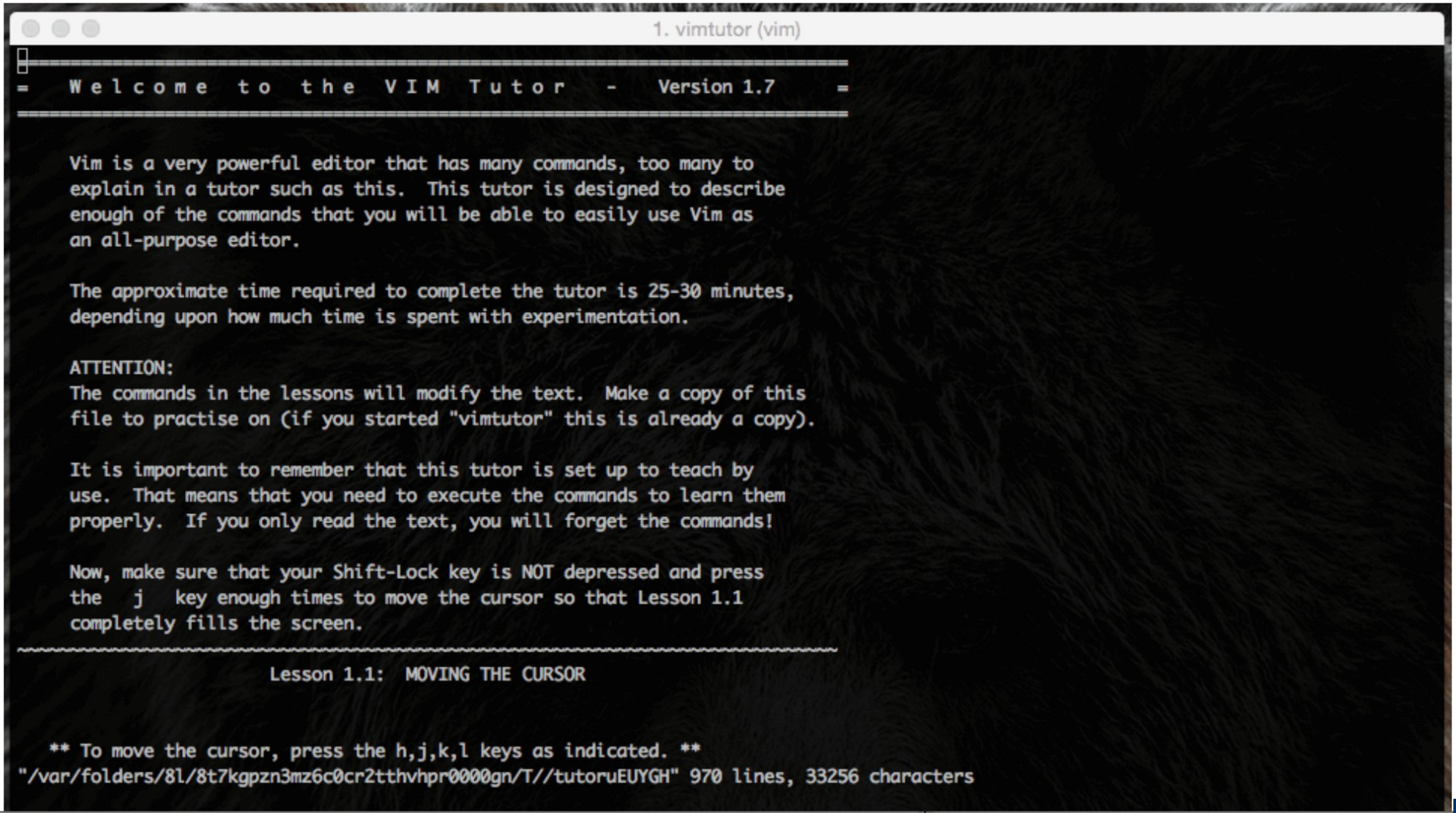
```
$ vimtutor
```

VIM Tutorial


1. vimtutor (vim)

```
===============================================================================
=    W e l c o m e   t o   t h e   V I M   T u t o r    -    Version 1.7      =
===============================================================================

     Vim is a very powerful editor that has many commands, too many to
     explain in a tutor such as this.  This tutor is designed to describe
     enough of the commands that you will be able to easily use Vim as
     an all-purpose editor.

     The approximate time required to complete the tutor is 25-30 minutes,
     depending upon how much time is spent with experimentation.

     ATTENTION:
     The commands in the lessons will modify the text.  Make a copy of this
     file to practise on (if you started "vimtutor" this is already a copy).

     It is important to remember that this tutor is set up to teach by
     use.  That means that you need to execute the commands to learn them
     properly.  If you only read the text, you will forget the commands!

     Now, make sure that your Shift-Lock key is NOT depressed and press
     the   j   key enough times to move the cursor so that Lesson 1.1
     completely fills the screen.
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
                        Lesson 1.1:  MOVING THE CURSOR


   ** To move the cursor, press the h,j,k,l keys as indicated. **
"/var/folders/8l/8t7kgpzn3mz6c0cr2tthvhpr0000gn/T//tutoruEUYGH" 970 lines, 33256 characters
```

VIM Tutorial



```
===============================================================================
=    W e l c o m e   t o   t h e   V I M   T u t o r    -    Version 1.7      =
===============================================================================

     Vim is a very powerful editor that has many commands, too many to
     explain in a tutor such as this.  This tutor is designed to describe
     enough of the commands that you will be able to easily use Vim as
     an all-purpose editor.

     The approximate time required to complete the tutor is 25-30 minutes,
     depending upon how much time is spent with experimentation.

     ATTENTION:
     The commands in the lessons will modify the text.  Make a copy of this
     file to practise on (if you started "vimtutor" this is already a copy).

     It is important to remember that this tutor is set up to teach by
     use.  That means that you need to execute the commands to learn them
     properly.  If you only read the text, you will forget the commands!

     Now, make sure that your Shift-Lock key is NOT depressed and press
     the   j   key enough times to move the cursor so that Lesson 1.1
     completely fills the screen.
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
                        Lesson 1.1:  MOVING THE CURSOR


   ** To move the cursor, press the h,j,k,l keys as indicated. **
"/var/folders/8l/8t7kgpzn3mz6c0cr2tthvhpr0000gn/T//tutoruEUYGH" 970 lines, 33256 characters
```

VIM – Moving the cursor

From the earlier example, you were probably using the arrow keys to navigate around.

That's perfectly okay, but it's recommended that you navigate a different way and actually not with the arrow keys.

- This way may be unnatural or weird at first, but it's recommended to use these keys instead:
  - `h` - Left
  - `k` - Up
  - `l` - Right
  - `j` - Down

Text editing - Deletion

- It's one thing to delete text from **Insert Mode**, but you can also delete text from **Command Mode**.

  To do that, click the editor and hit ESC to enter **Command Mode**. Next, navigate to any letter you want to delete and hit:

  x

- Wondering why you just won't enter **Insert Mode** to delete characters?

  You can always do that, but you'll see from future examples that deleting text from **Command Mode** is more powerful and much quicker.

Text editing - Insertion

- Text editing simply requires you that you enter **Insert Mode**. But there's some other methods to do this that can help speed things up and save you some keystrokes. So,
  - **Insterting**

    i
  - **Appending**
    - a

  This puts the cursor **after the current position**.

Text editing - Instertion

Open commands:

- This puts the cursor **below the line**:

  o

- And this puts the cursor **above the line**:

  O

Commands

- Commands are where the true power and efficiency come from Vim. It takes time to start using them, but they all follow a similar pattern. **It's a good idea to learn how commands work, not memorize commands.** Commands are broken down into these parts:
  - Operator
  - Numbers
  - Motions

When to put together, the Vim Command will look something like this:

```
[OPERATOR][NUMBER][MOTION]
```

Operators

Operators are actions. These are like **verbs** of a sentence when "speaking Vim".

Here's a list of common operators:

d - Delete (acts like a "cut" command though)

c - Change

y - Yank

p - Insert last deleted text after cursor (put command)

r - Replace

Motions

Motions provide **context** to your Operators. These execute the action in a particular way.

Here's a list of common motions:

w - Until the start of the next word, **EXCLUDING** its first character.

e - To the end of the current word, **INCLUDING** the last character.

$ - To the end of the line, **INCLUDING** the last character.

Motions (cont.)

And some additional others:

`w` - Forward by word

`b` - Backward by word

`)` - Beginning of next sentence

`(` - Beginning of current sentence

`}` - Beginning of next paragraph

`{` - Beginning of current paragraph

`]` - Beggining of next sect

`[` - Begginning of current section

`H` - Top line of screen

`L` - Last line of screen

Counts

Counts are optional and simply let you put a multiplier to your command and motion. You'll see how these work in the examples below.

In time you'll learn more and more of these and get quicker and quicker.

Let's go over some examples to demo how these work to together.

Examples

- **Deleting a word:**

Navigate to beginning of the word in the editor below and enter this command. If you're in the middle it will stop at where the word ends:

```
dw
```

- Deleting to end of line:

This will delete everything until the end of the line. Move your cursor to the beginning of a line and enter this command:

```
d$
```

Examples

- Example of a count.

This will run the command twice and deleting two lines after the cursor position:

d2$

- Deleting a line is a super common task. Vim has a shortcut built in for this. For example, to quickly delete a line you can always just do **dd** instead.
- Deleting 4 words:

Here's a command for deleting four sequential words:

d4w

UNDO Command

With all these commands, there's a good chance you might mess up once or twice.

This is totally normal and okay. You can quickly undo a command with:

```
u
```

Page Navigation

- Move to bottom of file:

  G

- Move to start of file:

  gg

- Jump to a specific line with:

  <num>+G

  Example:

  - 3+G

Searching

- Search a page after cursor position:

```
/cats
```

- Search a page before cursor position:

```
?dogs
```

- Go to next match

```
n
```

- Go to previous match

```
N
```

Search and Replace

- Searching and jumping around the page is one thing, but maybe you want to change all words of `cat` to the word `dog`. This is really easy with Vim.
- Find and replace:

```
:s/cat/dog
```

- Find and replace all:

```
:s/cat/dog/g
```

Execute and external command

With Vim, the commands aren't just limited to the Vim syntax/language of operators and motions.

You can execute external commands as you normally would from the command line inside of the editor. All you need to do is start the command with an *exclamation mark*.

Here's an example to list all files:

```
:!ls
```

Q/A ?

Thank you !