

# **Vim Hands-On Tutorial**

## **Computer Science Department**

<b>INTRODUCTION:</b> .....	<b>3</b>
TUTORIAL REQUIREMENTS: .....	3
TUTORIAL CONVENTIONS: .....	3
ABOUT VIM .....	3
<b>EDITING BASICS:</b> .....	<b>4</b>
INVOKING VIM.....	4
VIM MODES: COMMAND AND INSERT .....	4
OPENING EXISTING FILES AND SAVING (OR NOT).....	5
MOVING AROUND.....	8
OOPS! (UNDOING MISTAKES) .....	8
<b>TEXT SEARCHING:</b> .....	<b>10</b>
SEARCH BY PATTERN.....	10
SEARCHING BY LINE NUMBER .....	12
SUBSTITUTION: .....	13
<b>MANIPULATING TEXT:</b> .....	<b>14</b>
COPYING.....	14
DELETING/CUTTING.....	14
PASTING.....	14
<b>RECOVERING FROM A SYSTEM CRASH:</b> .....	<b>15</b>
<b>CUSTOMIZING VIM:</b> .....	<b>15</b>
<b>REFERENCES</b> .....	<b>16</b>
<b>EVALUATION</b> .....	<b>17</b>

## Introduction:

### ***Tutorial Requirements:***

- Computer Science Department Linux account
- Familiarity with standard feature and functions of a text editor
- Previous experience working in the Linux environment, attended a session of the CS Department Hands-on Linux Tutorial, or completed at least one CS course that required the use of the Linux environment.

### ***Tutorial Conventions:***

- Vim Commands are in *italics*.
- Tutorial Examples Commands are in **bold**.
- Keyboard strokes are in angle brackets, i.e. <enter> to strike the enter key

### ***About Vim***

Vim is a text editor that stands for “Vi Improved”. As the name suggests, it is upwards compatible to Vi and it can be used to edit all kinds of plain text documents. It's typically used to edit program source files and Linux/Unix system configuration files. It also happens to be installed by default on almost every Unix or Unix-clone (i.e. Linux) platform.[1]

Some improvements that Vim has over Vi are multi-level undo, multi windows and buffers, syntax highlighting, command line editing, filename completion, online help, visual selection, etc.. See *:help vi\_diff.txt* for a summary of the differences between Vim and Vi. [1]

While running Vim, help can be obtained from the on-line help system, with the *:help* command.

Most often Vim is started to edit a single file with the command:

```
vim filename
```

More generally, Vim is started with:

```
vim [options][filelist]
```

A list of options can be obtained by reading the manual pages on Vim using the command:

```
man vim
```

There is an interactive Vim tutorial program that you can work through by using the command:

```
vimtutor
```

## Editing Basics:

### Invoking Vim

To start Vim on the command line type:

```
vim <Enter>
```

You will see the welcome screen in the Vim program appear:

```
ms39856@dh231m11:~
File Edit View Terminal Go Help

VIM - Vi IMproved

version 6.2.98
by Bram Moolenaar et al.
Vim is open source and freely distributable

Help poor children in Uganda!
type :help iccf<Enter> for information

type :q<Enter> to exit
type :help<Enter> or <F1> for on-line help
type :help version6<Enter> for version info

0,0-1 All
```

### Vim Modes: Command and Insert

There are two primary modes in Vim: Command and Insert. Vim starts in Command mode. You cannot insert text while in Command mode.

To switch from Command mode to Insert mode type:

```
i
```

The word 'insert' should appear at the bottom of the screen indicating you are in Insert mode.

Now you can insert text at the cursor position. Type the following traditional Hello World program into the buffer:

```
#include <iostream>
using namespace std;

int main()
{
    cout << "Hello World\n";
    return 0;
}
```

```
ms39856@dh231m11:~
File Edit View Terminal Go Help

VIM - Vi IMproved

version 6.2.98
by Bram Moolenaar et al.
Vim is open source and freely distributable

Help poor children in Uganda!
type :help iccf<Enter> for information

type :q<Enter> to exit
type :help<Enter> or <F1> for on-line help
type :help version6<Enter> for version info

-- INSERT --
0,1 All
```







## Moving Around

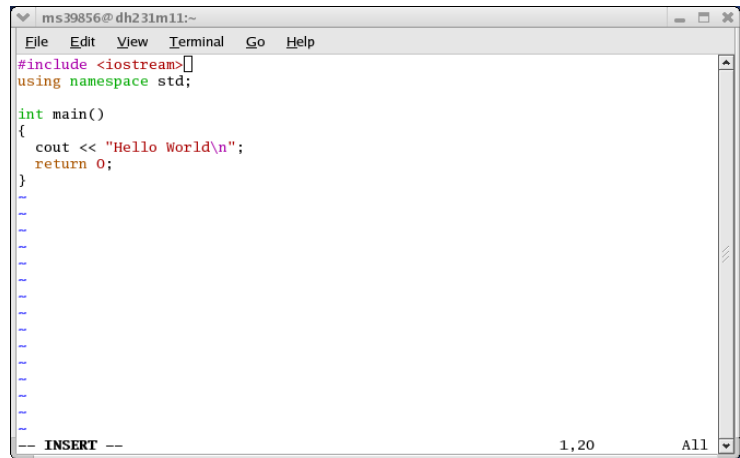
As you may have already found out, when you are in Insert mode you can use the arrow keys to move left, right, up, or down. There are a few tricks for moving through text in Command mode that can be helpful.

Now type:

Do not need! **vim hello.cpp** <Enter>

To insert (append) text at the end of a line type **A** in Command mode. This positions the cursor at the end of the line and switches Vim to Insert mode. Now move your cursor to the top hello.cpp and type:

**A**



```

ms39856@dh231m11:~
File Edit View Terminal Go Help
#include <iostream>[]
using namespace std;

int main()
{
  cout << "Hello World\n";
  return 0;
}
-- INSERT --
1,20 All

```

To insert text at the beginning of a line, type **I** in Command mode. This positions the cursor at the beginning of the line and switches Vim to Insert mode. Go back to Command mode by pressing:

<Esc>

Then type:

**I**

Now press:

<Esc>

Now exit your hello.cpp file by typing:

**:q** <enter>

Now type:

**vim /etc/profile** <Enter>

To scroll forward through a screen of text type

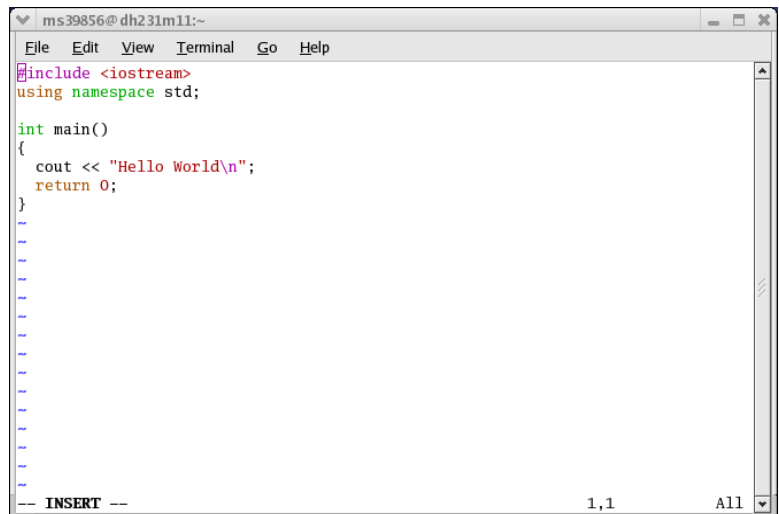
<Ctrl> **f**

To scroll backward through a screen of text type

<Ctrl> **b**

Now close the /etc/profile file by typing:

**:q** <Enter>



```

ms39856@dh231m11:~
File Edit View Terminal Go Help
#include <iostream>
using namespace std;

int main()
{
  cout << "Hello World\n";
  return 0;
}
-- INSERT --
1,1 All

```

## Oops! (Undoing Mistakes)

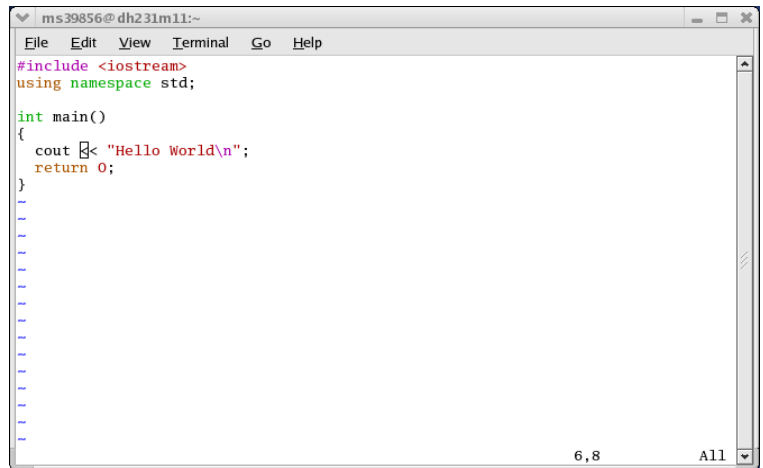
Like almost every good text editor, there is an undo feature to correct past mistakes in editing your file. Vim keeps a history of your edits and can undo multiple edits.



To undo all edits on a single line in the file type:

**U**

This command will only work if your cursor is still on the line you wish to undo. Once your cursor has moved off the line you cannot undo the line with the *U* command.



```
ms39856@dh231m11:~
File Edit View Terminal Go Help
#include <iostream>
using namespace std;

int main()
{
  cout << "Hello World\n";
  return 0;
}
6,8 All
```

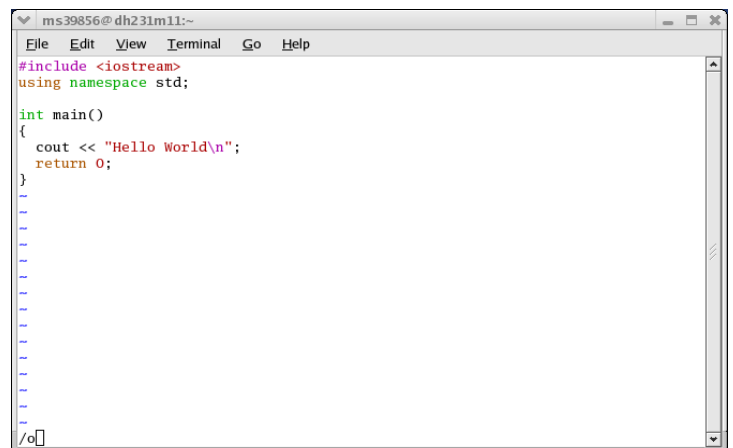
## Text Searching:

You can search for any string in Vim using a variety of search commands while in Command mode:

### *Search by Pattern*

To search forward for a pattern o type:

**/o**

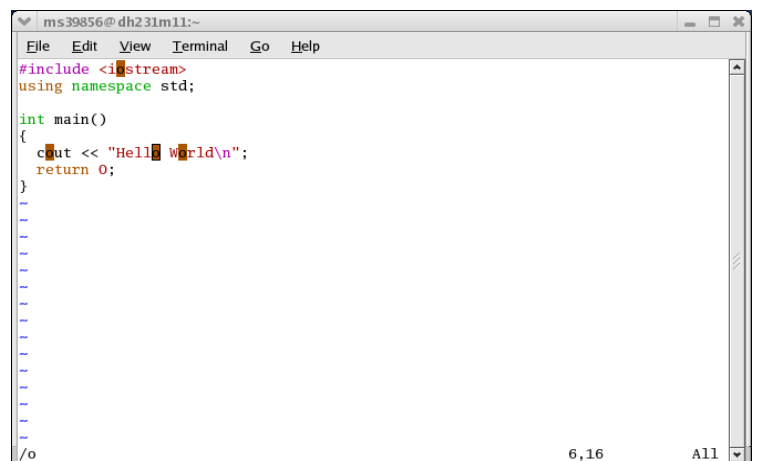


```
ms39856@dh231m11:~
File Edit View Terminal Go Help
#include <iostream>
using namespace std;

int main()
{
  cout << "Hello World\n";
  return 0;
}
/o
```

To search for the next instance of o type:

**n**



```
ms39856@dh231m11:~
File Edit View Terminal Go Help
#include <iostream>
using namespace std;

int main()
{
  cout << "Hello World\n";
  return 0;
}
6,16 All
```

To search for the previous instance of o type:

**N**

A screenshot of a code editor window titled "ms39856@ dh231m11:~". The editor contains the following C++ code:

```
#include <iostream>
using namespace std;

int main()
{
    cout << "Hello World\n";
    return 0;
}
```

The search bar at the bottom left shows "?o". The status bar at the bottom right indicates "6,4" and "All".

To search backward for a pattern e type:

**?e**

A screenshot of a code editor window titled "ms39856@ dh231m11:~". The editor contains the following C++ code:

```
#include <iostream>
using namespace std;

int main()
{
    cout << "Hello World\n";
    return 0;
}
```

The search bar at the bottom left shows "?e". The status bar at the bottom right indicates "2,15" and "All".

To search for the next instance of e type:

**n**

A screenshot of a code editor window titled "ms39856@ dh231m11:~". The editor contains the following C++ code:

```
#include <iostream>
using namespace std;

int main()
{
    cout << "Hello World\n";
    return 0;
}
```

The search bar at the bottom left shows "?e". The status bar at the bottom right indicates "2,10" and "All".

To search for the previous instance of e type:

**N**

```
ms39856@dh231m11:~
File Edit View Terminal Go Help
#include <iostream>
using namespace std;

int main()
{
    cout << "Hello World\n";
    return 0;
}

/e 2,15 All
```

## Searching by Line Number

Compilers generally display error messages using line numbers. So being able to find specific lines based on the line number is a very useful feature.

To find the line number of your current cursor position, press and hold:

Then type:

**<Ctrl> g**

```
ms39856@dh231m11:~
File Edit View Terminal Go Help
#include <iostream>
using namespace std;

int main()
{
    cout << "Hello World\n";
    return 0;
}

"hello.cpp" [Modified] 8 lines --25%-- 2,15 All
```

You should see the filename, line position, document location percent, and column number at the bottom of the screen.

To move to a specific line in the file, type the line number you want followed by **G** in Command mode. For example:

To move the cursor to line 44 you would type **44G**. Move your cursor to the top of hello.cpp and type:

**5G**

This should have moved your cursor to the cout statement in hello.cpp.

```
ms39856@dh231m11:~
File Edit View Terminal Go Help
#include <iostream>
using namespace std;

int main()
{
    cout << "Hello World\n";
    return 0;
}

"hello.cpp" [Modified] 8 lines --25%-- 5,1 All
```







## References

1. Moolenarr Bram, Thompson Tim, Andrews Tony, Walter G. R. Vim Manual Page. [Http://www.vimvim.org](http://www.vimvim.org). February 22 2002.
2. Lamb Linda, Robbins Arnold. Learning the vi Editor (6<sup>th</sup> Edition). O'Reilly. November 1998.

# Evaluation

**The intention of this tutorial is to familiarize the participant with vim editor.**

**Please leave your comments regarding the tutorial and indicate any areas improvement you feel should to be made.**

**Finally, thanks for coming and we hope this tutorial has proved useful and beneficial.**

**Comments:**

---

---

---

---

---

---

---

---

---

---