Vim Hands-On Tutorial
Computer Science Department
<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>INTRODUCTION</td>
<td>3</td>
</tr>
<tr>
<td>TUTORIAL REQUIREMENTS:</td>
<td>3</td>
</tr>
<tr>
<td>TUTORIAL CONVENTIONS:</td>
<td>3</td>
</tr>
<tr>
<td>ABOUT VIM</td>
<td>3</td>
</tr>
<tr>
<td>EDITING BASICS:</td>
<td>4</td>
</tr>
<tr>
<td>INVOKEING VIM</td>
<td>4</td>
</tr>
<tr>
<td>VIM MODES: Command and Insert</td>
<td>4</td>
</tr>
<tr>
<td>OPENING EXISTING FILES AND SAVING (OR NOT)</td>
<td>5</td>
</tr>
<tr>
<td>MOVING AROUND</td>
<td>8</td>
</tr>
<tr>
<td>OOPS! (UNDOING MISTAKES)</td>
<td>8</td>
</tr>
<tr>
<td>TEXT SEARCHING:</td>
<td>10</td>
</tr>
<tr>
<td>SEARCH BY PATTERN</td>
<td>10</td>
</tr>
<tr>
<td>SEARCHING BY LINE NUMBER</td>
<td>12</td>
</tr>
<tr>
<td>SUBSTITUTION:</td>
<td>13</td>
</tr>
<tr>
<td>MANIPULATING TEXT:</td>
<td>14</td>
</tr>
<tr>
<td>COPYING</td>
<td>14</td>
</tr>
<tr>
<td>DELETING/CUTTING</td>
<td>14</td>
</tr>
<tr>
<td>PASTING</td>
<td>14</td>
</tr>
<tr>
<td>RECOVERING FROM A SYSTEM CRASH:</td>
<td>15</td>
</tr>
<tr>
<td>CUSTOMIZING VIM</td>
<td>15</td>
</tr>
<tr>
<td>REFERENCES</td>
<td>16</td>
</tr>
<tr>
<td>EVALUATION</td>
<td>17</td>
</tr>
</tbody>
</table>
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:

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;
}
```

Rev. 10/25/04
After typing the Hello World program we are ready to switch from Insert Mode to Command mode. To enter Command Mode press:

\(<\text{Esc}>\)

Notice that the word Insert is not at the bottom of the screen. We are now in command mode.

---

**Opening Existing Files and Saving (or Not)**

In order to save our work and exit the vim program we type:

\(\text{:wq hello.cpp}\)

Then press \(<\text{Enter}>\)

This should return you to your bash shell prompt. From here you can see the file that vi created by using the \(ls\) command.

To open hello.cpp in ViM from the bash shell prompt we type:

\(\text{vim hello.cpp}\)

Now if we want to quit without saving we type:

\(\text{:q}\)

Then press:  \(<\text{Enter}>\)
If the file hello.cpp has been edited then Vim will not let you exit without saving.

Open hello.cpp in Vim by typing:

```
vim hello.cpp
```

Then type:

```
i
```

Now type:

```
change
```

Then press: `<Esc>`

Then type:

```
:q
```

Then press: `<Enter>`

If you want to exit Vim after editing hello.cpp then you have to do a forced quit by typing:

```
:q!
```

To insert another file into the current file at the current cursor position perform the following steps.

Type:

```
vim hello.cpp
```

First move the cursor to the bottom of the hello.cpp file and type:

```
:r .bash_profile
```
Then press: `<Enter>`

Now, if we edit the file and then want to remove all the edits we have made we type:

`:e!`

As you can see here, the `.bash_profile` file we inserted into our `hello.cpp` file is now gone.
**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:

```
 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
```

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
```

**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.

Rev. 10/25/04
Now type:

    vim hello.cpp  <Enter>

Go into Insert mode by typing:

    i

Then type:

    change

Press <Esc>

To undo your last edit type:

    u

For every consecutive u typed in command mode another of your last edits is undone.

Enter insert mode by typing:

    i

Go to your cout statement and change it to:

    cout >> "World Hello/n"

Press: <Esc>
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.

---

**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
```

To search for the next instance of `o` type:

```
/n
```
To search for the previous instance of o type:

N

To search backward for a pattern e type:

?e

To search for the next instance of e type:

n
To search for the previous instance of e type:

N

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:

\(<\text{Ctrl}>g\)

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.

Rev. 10/25/04
Substitution:

To perform a global substitution that will replace every instance of e to T type:

`:%s/e/T/g`

This will replace all instances of e with T no questions asked.

If you want to do a global substitution with confirmation on each instance of T with e then type:

`:%s/T/e/gc`

This will prompt you for every instance of T and ask if you want to substitute it for e.

Now type:

`y`

To quit type:

`q`
Manipulating Text:

**Copying**

The command to copy a line text is `yy` in Command mode. To copy multiple lines from the current cursor position enter the number of lines before `yy`. For example:

To copy 12 lines: `12yy`

To copy 12 lines: `12yy`

Move your cursor to the main function of the hello.cpp and type:

```
5yy
```

The number of lines you are copying should be displayed at the bottom of the screen as yanked lines.

**Deleting/Cutting**

Cutting lines works exactly like copying lines except replace the use “`yy`” with “`dd`”.

**Pasting**

To paste either copied or cut lines type `p` in command mode. This will paste the lines under the current position of the cursor.

Move your cursor to the bottom of the hello.cpp and type:

```
p
```

The number of lines pasted should be displayed at the bottom of the screen.

Now we will revert hello.cpp to it's original content and quit Vim by typing:

```
:e! <Enter>
```

Next type:

```
:q <Enter>
```

Rev. 10/25/04
Recovering from a system crash:

Like most editors, if the system crashes or the secure shell connection gets disconnected there is a way to recover the edit buffer for the file you were last editing. To recover the edit buffer type `vim -r hello.cpp` at the bash shell prompt.

You should see the message displayed in the figure at the right after recovering the edit buffer.

Customizing Vim:

You can set Vim options and customize Vim’s behavior by creating and editing a `.vimrc` file in your home directory. Here is an example of a nice `.vimrc` to assist the writing of programs in C++ or C.

The `nocp` option turns off strict vi compatibility. The `incsearch` option turns on incremental searching. The `number` option displays the line number to the left of each line and the `showmatch` option matches brackets `{}` after closing a block of code. The `terse` option turns off the message displayed when a search has wrapped around a file. The settings for `cinoptions`, `cinwords`, and `formatoptions` differ from the defaults. The result is to produce a fairly strict “K&R” C formatting style.[2]

For more information on all the different options that can be set type `:help option-summary` in Command mode.

Rev. 10/25/04
References

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:
__________________________________________________________________________________________
__________________________________________________________________________________________
__________________________________________________________________________________________
__________________________________________________________________________________________
__________________________________________________________________________________________
__________________________________________________________________________________________
__________________________________________________________________________________________
__________________________________________________________________________________________