

Vi - the standard UNIX Editor

Modes in vi

There are three basic modes of vi:

Command mode

This is the default when you enter vi. In command mode, most letters, or short sequences of letters, that you type will be interpreted as commands, *without explicitly pressing Enter*. If you press **Esc** when you're in command mode, your terminal will beep at you. This is a very good way to tell when you're in command mode.

Insert mode

In insert mode, whatever you type is inserted in the file at the cursor position. Type **a** (lowercase letter a, for *append*) to enter insert mode from command mode; press **Esc** to end insert mode, and return to command mode.

Line mode

Use line mode to enter line oriented commands. In command mode, type a colon (:). Your cursor moves to the bottom of the screen, by a colon prompt. Type a line mode command, then press **Enter**. Any sensible command from the UNIX line editor ex will work, and a few are good to know about. These commands are indicated in this handout by a colon in front of the command. Each time you use a line mode command, you must type a colon to enter line mode, then type the command by the colon prompt at the bottom of the screen, then press **Enter** when you finish typing the command. (The search commands starting with / and ? work similarly.)

Starting vi and Saving Files

<i>Starting vi:</i>	
vi filename	start editing <i>filename</i> , create it if necessary
<i>Saving the file you're working on and/or leaving vi:</i>	
:wq	write the file to disk and quit
:q!	quit without saving any changes
:w! newfile	write all lines from the entire current file into the file ' <i>newfile</i> ', overwriting any existing <i>newfile</i>
:n,m w!	write the lines from <i>n</i> to <i>m</i> , inclusive, into the file <i>newfile</i> , overwriting

<i>newfile</i>	any existing <i>newfile</i>
----------------	-----------------------------

Moving the Cursor

Many commands take number prefixes; for example **5w** moves to the right by 5 words.

<i>Type:</i>	<i>To Move To:</i>
h	one space to the left (also try left arrow)
j	one line down (also try down arrow)
k	one line up (also try up arrow)
l	one space to the right (also try right arrow)
\$	end of current line
^	beginning of current line
Enter	beginning first word on the next line
G	end of file
:n	line <i>n</i>
w	beginning of next word
e	end of next word
b	beginning of previous word
Ctrl-b	one page up
Ctrl-f	one page down
%	the matching (,), [,], {, or } (Press % with your cursor on one of these characters to move your cursor its mate.)

Searching for Text

Type:	To:
<i>/string</i>	search down for <i>string</i>
<i>?string</i>	search up for <i>string</i>
n	repeat last search from present position

Inserting Text

Type:	To:
a	append starting right of cursor
A	append at the end of the current line
i	insert starting left of cursor
I	insert at beginning of the current line
o	open line below cursor, then enter insert mode
O	open line above cursor, then enter insert mode
:r newfile	add the contents of the file <i>newfile</i> starting below the current line

Deleting Text

Type:	To:
x	delete single character; 5x deletes 5 characters
dw	delete word; 5dw deletes 5 words
dd	delete line; 5dd deletes ... well you get the idea!
cw	delete word, leaves you in insert mode (i.e. change word)
cc	change line -- delete line and start insert mode

s	change character -- delete character and start insert mode
D	delete from cursor to end of line
C	change from cursor to end of line -- delete and start insert mode
u	undo last change
U	undo all changes to current line
J	join current line with line that follows (press Enter in insert mode to split line)

Cutting and Pasting

<i>Type:</i>	<i>To:</i>
xp	transpose two characters (two commands, x followed by p)
yy	yank (i.e. copy) one line into a general buffer (5yy to yank 5 lines)
"ayy	yank into the buffer named <i>a</i>
P	put the general buffer back before the current line
"aP	put from buffer <i>a</i> before current line
p	put the general buffer back after the current line
"ap	put from buffer <i>a</i> after the current line

Using the Pico Editor in UNIX

What is Pico?

Pico is a file editor used on UNIX systems. It comes with pine, where it is used by default to compose new messages, but it can also be used as a stand-alone editor. It is invoked by the command **pico *filename***, where *filename* is replaced with the name of your file.

Editing commands are invoked by control key sequences (i.e. pressing the control key together with another key or, on a Macintosh, pressing the Escape key twice, then another key). Several menu-like status lines at the bottom of the screen show which commands are active at any given time. Key strokes that do not constitute control key sequences are entered as text at the current cursor position.

Cursor Movement:

You can move your cursor by using the arrow keys; you may also use any of the following editing commands:

Ctrl-a Beginning of current line

Ctrl-e End of current line

Ctrl-v Forward one screen

Ctrl-y Backward one screen

Saving a File:

You can save the file you are working on by using the **Ctrl-o** (write out) command. If you exit Pico by using the **Ctrl-x** (exit) command, you will be given a chance to save the file.

Deleting Text:

Ctrl-d Delete the character the cursor is on

Ctrl-e Delete to end of current line

Ctrl-k Delete the line the cursor is on

Ctrl-u Restore the last line that was deleted

Cutting and Pasting:

You can mark text for cutting and pasting by using the **Ctrl-^** command. After entering this command, move the cursor to highlight the text you wish to cut. Then use **Ctrl-u** to paste it.

Searching:

You can search for a given text by using the **Ctrl-w** command. The search is NOT case sensitive. After entering the command, you will be prompted for the text for which you want to search. If you press Enter at this point, Pico will repeat the last search. All searches start at the current cursor position and wrap around to the beginning of the file.

To move to the last line in the file, type **Ctrl-w** (where is), **Ctrl-v** (last line). To move to the first line of the file type **Ctrl-w** (Whereis), **Ctrl-y** (first line).

Justifying Text:

The justify command, **Ctrl-j**, will remove extra white space and new lines to make lines as equal in length as possible. It operates on the paragraph where the cursor is currently positioned. Pico defines a paragraph as text separated by a blank line or a line starting with a blank or tab.

By default, Pico 'wraps' lines at 80 characters. It will not break text in the middle of a word, however, but will go back to the most previous white space. This default can be overridden by starting Pico with the **-w** option. When this is in effect, lines too long to be displayed on the screen will have a '\$' in the column furthest to the right.

Spell Checking:

Pico includes an elementary spell check function which can be invoked with **Ctrl-t**. It will flag words it cannot find and give you a chance to edit them, but will not suggest a correct spelling or allow you to customize the dictionary. You can also use **ispell**, a more sophisticated spell checker, on any file. At the command prompt, type **ispell filename** , where "filename" is the name of the file you want to spell check.

Help:

Online help is available with the **Ctrl-g** (help) command. It invokes a short but fairly complete document that you can page through. The command line options are discussed more thoroughly in the man page. To see the man page for Pico, enter **man pico** at the command prompt.

Command Line Options:

The following command line options are available for use when Pico is invoked (e.g. **pico -w filename**):

-w Disables word wrapping

+x (x is an integer) Starts the cursor x lines into the buffer

- (x is an integer) Checks your mailbox every x seconds and notifies you if you have **nx** new mail.

Miscellaneous:

Ctrl-c will report the current cursor position in the buffer by both line number and character at the bottom of the screen.

Files are read into Pico with **Ctrl-r** and written with **Ctrl-o**. After either of these commands, you can enter **Ctrl-t** (list files) to invoke the file browser. The file browser will allow you to step through the directory structure, see a listing of all files in a given directory, copy files, rename files, and delete files.

Basic Computer Skills

Goals:

This paper introduces the basic skills needed to complete future assignments, including:

- logging into and navigating the Linux system
 - log in
 - change password
 - learn basic navigation/file system commands
 - create a text file
 - compile a text file
 - run a program
- logging into and navigating the Windows system
 - log in
 - change password
 - remotely access a CS Linux machine

Guided Activities:

The following step by step instructions will describe how to complete all of the operations listed in the goals section. It is to your benefit to try out the following activities, but feel free to discuss them with your peers.

A. Navigating the Linux system

1. If your Linux account still has the password assigned by root, it's a good idea to change it. For simplicity, you may want to use the same password you used for your Windows login.
 1. At the prompt, type in: `yppasswd`
 2. Follow the directions as they appear
2. Now that you've seen how to login, and plugged a big security hole your account, it's time to learn more about moving around the way file system. Linux uses a hierarchical system of folders to organize its files. Simply put, you have a structure of folders. Each folder may contain either files or other folders, or both.

When you log in, you're automatically brought to your home directory. To see what directory you're currently in, type **pwd** (print working directory) at the command prompt:

```
$ pwd
```

You will get:

```
/users/yzhang
```

**note: When you type `pwd`, you will get your user name in the place of `yzhang`.*

**note1: `$` is the default prompt on our Linux system, so you do not need to type it in. This is the way the computer tells you it's waiting for you to do something. Some other Linux systems use `>` or `%` as prompt.*

This directory is where you can store your personal files. The notation `/users/yzhang` denotes that there is a directory `yzhang` that is nested inside of another directory called `/users`. We would normally call `/users` a *parent directory* of `yzhang`, and call `yzhang` a *subdirectory* of `/users`.

To list the contents of the current directory, type `ls` (short for list). Try the following options:

```
$ ls
$ ls -l
$ ls -a
$ ls -al
```

`ls` gives the standard listing of the directory. `ls -l` will display additional information about each files. `ls -a` will show all files, including hidden files.

To make a new directory, you use the **`mkdir`** command (short for make directory). Try using the `mkdir` command to make a folder to store all of your future assignments:

```
$ mkdir CSCI1320
```

You can use `ls` to verify that the directory `CSCI1320` was created properly. Please note that file and folder names on a Linux system are case sensitive. Thus, `CSCI1320` is NOT the same as `csci1320`.

Let's change our current directory to be the new folder we just created. To accomplish this, we use the **`cd`** command (change directory).

```
$ cd CSCI1320
```

Now, if we type `pwd`, we get:

```
$ /users/yzhang/CSCI1320
```

If you type `ls`, you'll see that the directory is empty. However, there are some hidden directories that are automatically added to every folder. To display them, use:

```
$ ls -a
```

You should get the following output:

```
...
```

The `“.”` refers to the current directory. Thus, if we type:

```
$ cd .
$ pwd
```

We'll get the directory we started in. The `“..”` refers to the parent directory. Thus, if we started in `/users/yzhang/CSCI1320` and typed:

```
$ cd ..
```

```
$ pwd
```

we would get the /users/yzhang directory.

If you ever want to refer to your home directory, you can use the ~ symbol. Thus, to get to my /users/yzhang/CSCI1320 directory, I could use the following command:

```
$ cd ~/CSCI1320
```

If you ever wish to clear the screen, you can use the **clear** command:

```
$ clear
```

Two more commands, **cp** and **mv**, will be demonstrated after the introduction to the text editor that follows.

3. Now that now the basics of moving around the Linux file system, it's time to create a file of your own. Many people like **vi**, though there are many others who hate it with a vengeance. If you'd like to use vi, you are more than welcome to. I recommend <http://www.thomer.com/vi/vi.html> as a good place to get started. **Pico** is another text editor available on Linux. It is very easy to learn and more than adequate for the documents you will write in this class. Let's create a file called hello.c:

```
$ pico hello.c
```

You'll notice that your screen will change. You are now running the pico editor. Until you exit out of it (see the commands at the bottom of the screen for the commands to save, exit, etc), you cannot use any of the commands like cd, ls, etc. Pico is almost exactly like using notepad in Windows. All it does it allow you to store text in a file.

Type the following program into pico exactly as it is shown:

```
/* The traditional first program in honor of Dennis Ritchie who
   invented C at Bell Labs in 1972.
*/

#include <stdio.h>

int main(void)
{
    printf("Hello!\n");

    return 0;
}
```

Now, press ctrl and O at the same time (hit enter when prompted), then press ctrl and X. This saves your file, then exits pico.

You should see the command prompt once again. Now, you can enter ls to see the file you just created:

```
$ ls
```

This should return:
hello.c

Although this is a valid c program, it's not yet in a form that the computer can understand. To convert it, we need to use a *compiler*. To invoke the compiler on our system, type:

```
$ cc hello.c
```

When you do this, you'll either get errors, or be returned to the command prompt. If you get errors, read them and see if they tell you what you may have mistyped, or, if that fails, ask for help. If you see just the command prompt, your file has most likely been compiled correctly.

The compiler should have outputted a file called **a.out**. You can check by using ls. If it is not there, it has not been compiled correctly and you will need to recompile.

Once the file is produced, you can run the program using:

```
$ ./a.out
```

If it prints out "Hello!", Congratulations! You've just written a working program!

Now that we have some files to work with, let's try a few more commands for renaming, moving, copying, and deleting files.

The **mv** command is used both for moving files around and changing their names. For instance, try changing the name of the a.out file you created to hello:

```
$ mv a.out HELLO
```

This moves the source file, a.out, to the target file, HELLO. Typing ls will show that a.out is gone, and a file called hello with the same contents is in the current directory.

It can also be used for moving files or folders into different directories. Try moving a.out into CSCI1320's parent directory:

```
$ mv a.out ../
```

or

```
$ mv a.out ../a.out
```

To make a copy of a file, use the cp command. For instance:

```
$ cp hello.c hello1.c
```

will create a copy of hello.c called hello1.c

To delete a file, use the **rm** command:

```
$ rm hello1.c
```

will delete the file hello1.c

Deleting a directory is similar, except that the command is rmdir, and you must delete the contents of a directory before you delete that directory. Or you can delete a directory without deleting the contents of the directory in ahead:

```
$ rm -r CSCI1320
```

You should be carefully to use this command because it will delete all contents under CSCI1320 directory (I do not think you want to do this forever!).

There are some other commands that you may find helpful. First is the **man** command. man is the help system in Linux. To get the manual page on any command, which contains a description of how to the command, type:

```
$ man <command>
```

Thus, to get the manual page on the ls command, you would type:

```
$ man ls
```

To get out of the man page, hit the "q" key.

more command can help you see more of the text. Type:

```
$ more hello.c
```

history command shows a history of the commands you have typed with its history number beside. Type:

```
$ history
```

exit command allows you to exit from a program, shell or log you out of a Linux network. Try:

```
$ exit
```

On your own:

Try to complete the following tasks using what you have learned:

1. Under your home directory, create a directory called “foo” with a directory inside it called “bar”.
2. Copy your hello.c file into foo/bar
3. Rename the hello.c file in your foo/bar directory to howdy.c
4. Edit howdy.c and change the word “Hello!\n” to “Howdy!\n”
5. Compile howdy.c
6. Run the compiled file.

B. Navigating the Windows system

If you want to work on a CS Linux machine from a computer which has Windows Operating System, there is a program on the Windows machine you are logged into that allows you to work on the remote machine. This program is called Putty. You can download Putty from the department FTP web site:

<http://www.cs.trinity.edu/ftp/pub/winxp-software/>

and use it to connect to the Linux machines in the CS building from any other computer.

a. Logging in

At a Windows computer, hold down the ctrl, alt and delete keys simultaneously. This should bring up a login dialog. Your user name and password are the ones you use to login trinity domain.

Example: Yu Zhang with password 12345678 would have:

```
Username:  yzhang
Password:  12345678
```

b. Change your password

It’s pretty easy for someone else to get into your account right now. Since no one wants someone else getting in to their account, now would be a good time to change your password.

- i. Press the ctrl, alt, and delete keys at the same time.
- ii. Select the change password option.

iii. Type in your old password, and the new password. Please be careful in your choice of password; make it something easy to remember but hard for others to guess.

c. Remotely login to the Linux system:

- Open Putty – Click on All Programs -> Computer Science -> PuTTY -> PuTTY
- Enter the following Host Name: bianca.cs.trinity.edu
- Select the SSH protocol
- *note: at this time, you can save these settings so you don't have to type them in again. Just put some name in the "Saved Sessions" field, then hit the Save button.*
- Hit the "Open" button
- Use your username and **Linux** password when prompted. The password change in Windows does not affect you the password on the Linux machine.
- If you've logged in correctly, you'll see a message and a prompt.
- After the prompt, type in command **ssh** janus01. This command locally login you to a local Linux machine named janus01. janus01 is one of computers in classroom 228; you can ssh any computer in this classroom from janus01 to janus21.

If this is your first time login, you will be asked to generate a public key.

The message is like this:

The authenticity of host 'janus01 (131.194.71.151)' can't be established.

RSA key fingerprint is

e3:10:98:ad:a8:2b:f5:37:05:49:88:05:d9:44:b2:64. Are you sure you want to continue connecting (yes/no)?

You will type *yes*.

Next you will be asked for password. After you type in the **Linux** password, you will successful login janus01.

Summary of Linux Commands

<i>yppasswd</i>	change password
<i>pwd</i>	“print the working directory”. Tells you where you are in the directory structure.
<i>ls</i>	“list” the contents of the current directory. <i>ls -l</i> is for a long listing. <i>ls -a</i> lists all files including hidden files.
<i>mkdir <dirname></i>	“make directory”. Make a subdirectory of the current directory with the name <i><dirname></i> .
<i>cd <dirname></i>	“change directory” to <i><dirname></i> . <i>cd</i> by itself changes the current directory to the home directory.
<i>clear</i>	clear screen
<i>mv</i>	“move (rename)”. A somewhat different command with many options. For the moment, <i>mv <filename1> <filename2></i> will rename <i>filename1</i> to <i>filename2</i> .
<i>cp</i>	“copy”. A somewhat different command with many options. For the moment, <i>cp <filename1> <filename2></i> will copy the first file to the second file, creating a duplicate and overwriting <i><filename2></i> if it already existed.
<i>rm</i>	“remove”. Usage: <i>rm <filename></i> This command removes the file and puts it in the <i>junk</i> subdirectory of your home directory until you log out.
<i>man</i>	“manual” usage <i>man commandname</i> Displays the on line manual pages. Example: <i>man mh</i> will tell you about how to use the <i>mh</i> mail handler. <i>man</i> is one of the most important facilities that you will have at your disposal, although it admittedly takes some time to get used to the way that the manual pages are written. Get used to them quickly, for they will answer many questions without having to ask anyone.
<i>more</i>	see “more” of the text. Usage: <i>more <filename></i> . Allows you to go through a file one screen at a time.
<i>history</i>	Shows a history of the commands you have typed with its history number beside. This enables you to repeat long commands without having to repeat all of the typing. By typing <i>!<number></i> you can repeat a command without having to type the entire command. To repeat the last command, simply type <i>!!</i> . To repeat the last command that began with <i>ma</i> you can type <i>!ma</i> .
<i>ssh</i>	locally log in to a local UNIX machine or execute a command on a remote machine. Usage: <i>ssh machinename</i> to remotely log in or <i>ssh machinename command</i> to execute a command.