AI STUDIO

LINUX

Text Editors

AUGUST 2025

AI STUDIO

Overview of Text Editors in Linux

↘ At some point, you will need to manually edit text files. You might be composing an email offline, writing a script to be used for **bash** or other command interpreters, altering a system or application configuration file, or developing source code for a programming language such as C, Python or Java.

↘ Linux users and administrators may sidestep using a text editor, instead employing graphical utilities for creating and modifying system configuration files. However, this can be both more laborious than directly using a text editor and more limited in capability. In fact, word processing applications (including those that are part of common office application suites) are not really basic text editors; they add a lot of extra (usually invisible) formatting information that will probably render system administration configuration files unusable for their intended purpose. Thus, knowing how to confidently use one or more text editors is really an essential skill to have for Linux.

↘ By now, you have certainly realized Linux is packed with choices; when it comes to text editors, there are many choices, ranging from quite simple to very complex, including:

↘ nano

↘ gedit

↘ vi

↘ emacs

↘ In this section, we learn first about the **nano** and **gedit** editors, which are relatively simple and easy to learn, and then later the more complicated choices, **vi** and **emacs.**

↘ Another product that has gained in usage and popularity is Microsoft's Visual Studio Code, usually abbreviated as **code** when used in Linux. This is actually a fully featured integrated development environment and far from lightweight, but many new Linux users are already used to it from working in other operating systems. Installation details vary from one distribution to another and usually involve incorporating additional package repositories; we will not go into how to do this here.

AI STUDIO

Creating Files Without Using an Editor

↘ Sometimes, you may want to create a short file and not want to bother invoking a full text editor. Furthermore, doing so can be quite useful when used from within scripts, even when creating longer files. You will no doubt find yourself using this method when you start on the later chapters that cover shell scripting!

↘ If you want to create a file without using an editor, there are two standard ways to create one from the command line and fill it with content.

↘ The first is to use echo repeatedly:

```
$ echo line one > myfile
$ echo line two >> myfile
$ echo line three >> myfile
```

↘ Note that while a single greater-than sign (>) will send the output of a command to a file (and obliterate any already existing version of that file!), two of them (>>) will append the new output to an existing file.

The second way is to use cat combined with redirection:

```
$ cat << EOF > myfile
> line one
> line two
> line three
> EOF
$
```

↘ In this example, the string used to show the beginning and end of the process need not be EOF; it could be STOP or any other string not used in the content itself. Both techniques produce a file with the following lines in it:

```
line one
line two
line three
```

↘ and are extremely useful when employed by scripts.

AI STUDIO

nano

↘ nano is easy to use, and requires very little effort to learn. To open a file, type nano <filename> and press Enter. If the file does not exist, it will be created.

↘ nano provides a two line shortcut bar at the bottom of the screen that lists the available commands. Some of these commands are:

CTRL-G

Display the help screen.

CTRL-O

Write to a file.

CTRL-X

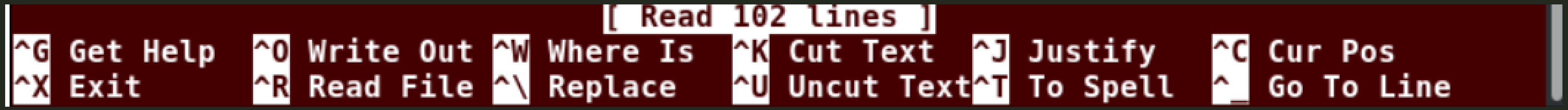
Exit a file.

CTRL-R

Insert contents from another file to the current buffer.

CTRL-C

Show cursor position.



```
                        [ Read 102 lines ]
^G Get Help   ^O Write Out  ^W Where Is   ^K Cut Text   ^J Justify    ^C Cur Pos
^X Exit       ^R Read File  ^\ Replace    ^U Uncut Text^T To Spell    ^_ Go To Line
```

AI STUDIO

Graphics Editors

Gedit

↘ gedit (pronounced 'g-edit') is a simple-to-use graphical editor that can only be run within a Graphical Desktop environment. It is visually quite similar to the Notepad text editor in Windows, but is actually far more capable and very configurable and hasa wealth of plugins available to extend its capabilities further.

↘ To open a new file find the program in your desktop's menu system, or from the command line type gedit <filename>. If the file does not exist, it will be created.

↘ Using gedit is pretty straightforward and does not require much training. Its interface is composed of quite familiar elements.

Visual Studio Code

↘ Visual Studio Code (code) is another simple-to-use graphical editor. It has a lot of functionality besides being used as a text editor. Here is a screenshot ofit working on the same file we have already shown with the other editors.

AI STUDIO

Text Editors

Vi and emacs

↘ Developers and administrators experienced in working on UNIX-like systems almost always use one of the two venerable editing options: vi and emacs. Both are present or easily available on all distributions and are completely compatible with the versions available on other operating systems.

↘ Both vi and emacs have a basic, purely text-based form that can run in a non-graphical environment. They also have one or more graphical interface forms with extended capabilities; these may be friendlier for a less experienced user. While vi and emacs can have significantly steep learning curves for new users, they are extremely efficient when one has learned how to use them.

↘ You need to be aware that fights among seasoned users over which editor is better can be quite intense and are often described as a holy war. It is clear, however, that there are many more users of vi than there are of emacs. Both editing programs are here to stay no matter what.

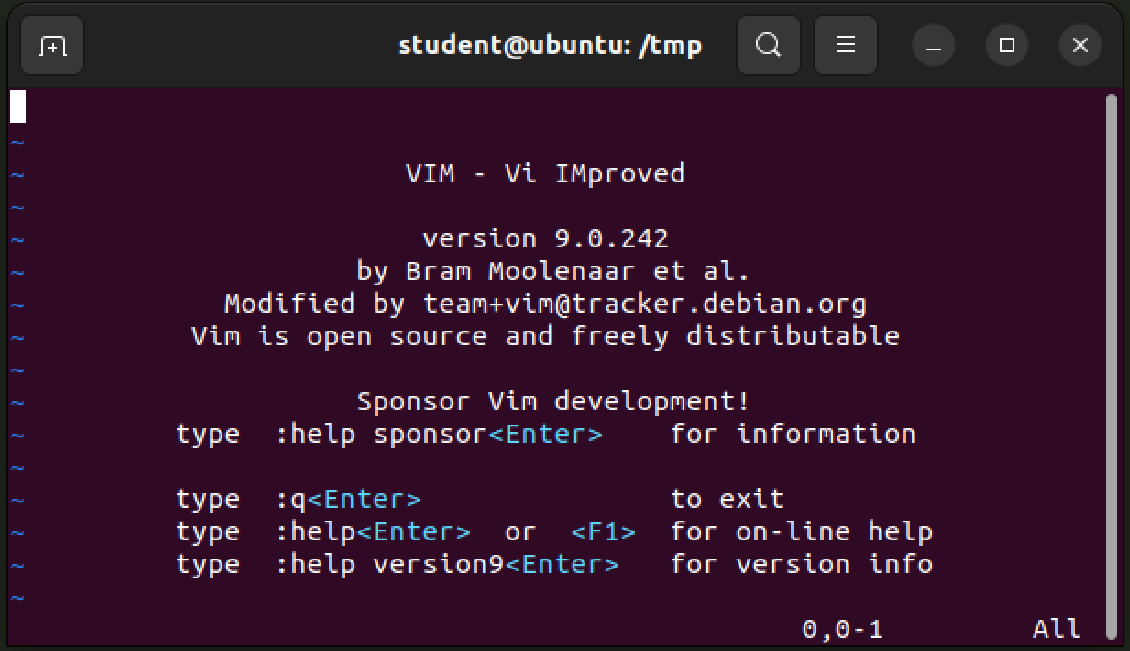
AI STUDIO

Introduction to vi

↘ Usually, the actual program installed on your system is vim, which stands for Vi IMproved and is aliased to the name vi. The name is pronounced as “vee-eye”.

↘ Even if you do not want to use vi, it is good to gain some familiarity with it: it is a standard tool installed on virtually all Linux distributions. Indeed, there may be times when there is no other editor available on the system.

↘ GNOME extends vi with a very graphical interface known as gvim and KDE offers kvim. Either of these may be easier to use at first. When using vi, all commands are entered through the keyboard. You do not need to keep moving your hands to use a pointer device such as a mouse or touchpad, unless you want to do so when using one of the graphical versions of the editor.

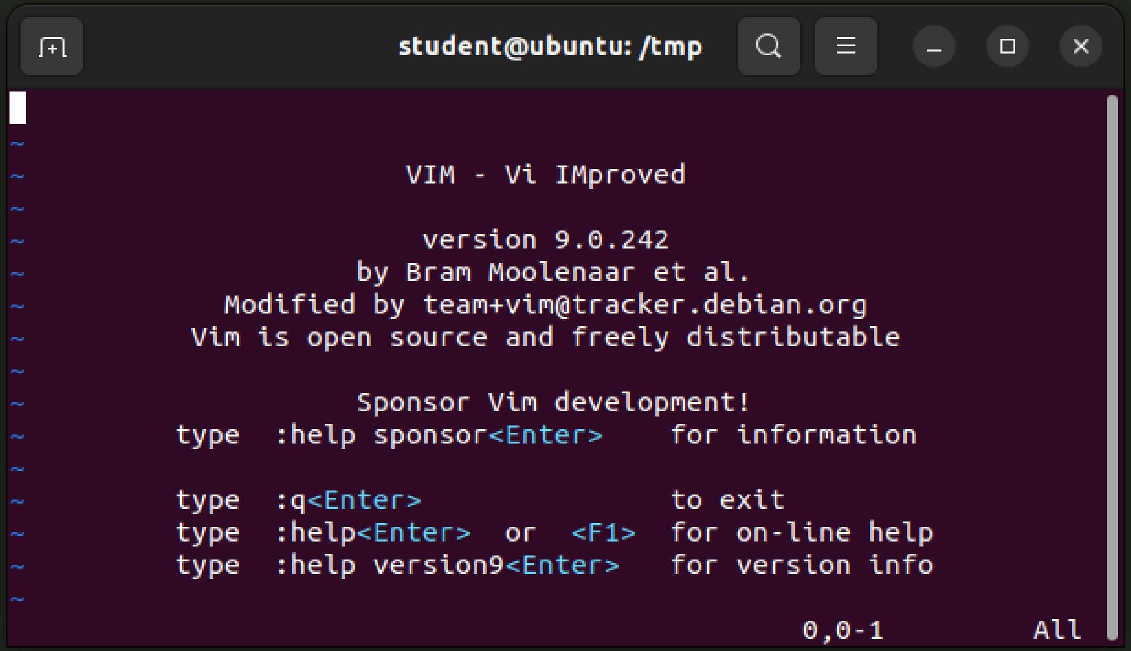


```
~
~                     VIM - Vi IMproved
~
~                      version 9.0.242
~                  by Bram Moolenaar et al.
~           Modified by team+vim@tracker.debian.org
~         Vim is open source and freely distributable
~
~                  Sponsor Vim development!
~        type  :help sponsor<Enter>    for information
~
~        type  :q<Enter>               to exit
~        type  :help<Enter>  or  <F1>  for on-line help
~        type  :help version9<Enter>   for version info
~
                                             0,0-1         All
```

AI STUDIO

vimtutor

↘ Typing vimtutor launches a short but very comprehensive tutorial for those who want to learn their first vi commands. Even though it provides only an introduction and just seven lessons, it has enough material to make you a very proficient vi user, because it covers a large number of commands. After learning these basic ones, you can look up new tricks to incorporate into your list of vi commands because there are always more optimal ways to do things in vi with less typing.



```
~
~                      VIM - Vi IMproved
~
~                       version 9.0.242
~                   by Bram Moolenaar et al.
~           Modified by team+vim@tracker.debian.org
~         Vim is open source and freely distributable
~
~                  Sponsor Vim development!
~        type  :help sponsor<Enter>    for information
~
~        type  :q<Enter>               to exit
~        type  :help<Enter>  or  <F1>  for on-line help
~        type  :help version9<Enter>   for version info
~
                                              0,0-1         All
```

AI STUDIO

Modes in vi

↘ vi provides three modes, as described in the table below. It is vital to not lose track of which mode you are in. Many keystrokes and commands behave quite differently in different modes.

MODE	FEATURE
Command	By default, vi starts in Command mode. Each key is an editor command. Keyboard strokes are interpreted as commands that can modify file contents.
Insert	Type **i** to switch to Insert mode from Command mode. Insert mode is used to enter (insert) text into a file. Insert mode is indicated by an “**? INSERT ?**” indicator at the bottom of the screen. Press **Esc** to exit Insert mode and return to Command mode.
Line	Type **:** to switch to the Line mode from Command mode. Each key is an external command, including operations such as writing the file contents to disk or exiting. Uses line editing commands inherited from older line editors. Most of these commands are actually no longer used. Some line editing commands are very powerful. Press **Esc** to exit Line mode and return to Command mode.

AI STUDIO

Working with files in vi

↘ The table describes the most important commands used to start, exit, read, and write files in vi. The ENTER key needs to be pressed after all of these commands.

COMMAND	USAGE
vi myfile	Start the editor and edit **myfile**
vi -r myfile	Start and edit **myfile** in recovery mode from a system crash
:r file2	Read in **file2** and insert at current position
:w	Write to the file
:w myfile	Write out to **myfile**
:w! file2	Overwrite **file2**
:x or :wq	Exit and write out modified file
:q	Quit
:q!	Quit even though modifications have not been saved

AI STUDIO

Changing Cursor Positions in vi

↘ The table describes the most important keystrokes used when changing cursor position in vi. Line mode commands (those following colon :) require the ENTER key to be pressed after the command is typed.

KEY	USAGE
arrow keys	To move up, down, left and right
j or **<ret>**	To move one line down
k	To move one line up
h or Backspace	To move one character left
l or Space	To move one character right
0	To move to beginning of line
$	To move to end of line
w	To move to beginning of next word
:0 or **1G**	To move to beginning of file
:n or nG	To move to line n
:$ or G	To move to last line in file
CTRL-F or **Page Down**	To move forward one page
CTRL-B or **Page Up**	To move backward one page
^l	To refresh and center screen

AI STUDIO

Searching for Text in vi

↘ The table describes the most important commands used when searching for text in vi. The ENTER key should be pressed after typing the search pattern.

COMMAND	USAGE
/pattern	Search forward for pattern
?pattern	Search backward for pattern

↘ The table describes the most important keystrokes used when searching for text in vi.

KEY	USAGE
n	Move to next occurrence of search pattern
N	Move to previous occurrence of search pattern

AI STUDIO

Working with Text in vi

↘ The table describes the most important keystrokes used when changing, adding, and deleting text in vi.

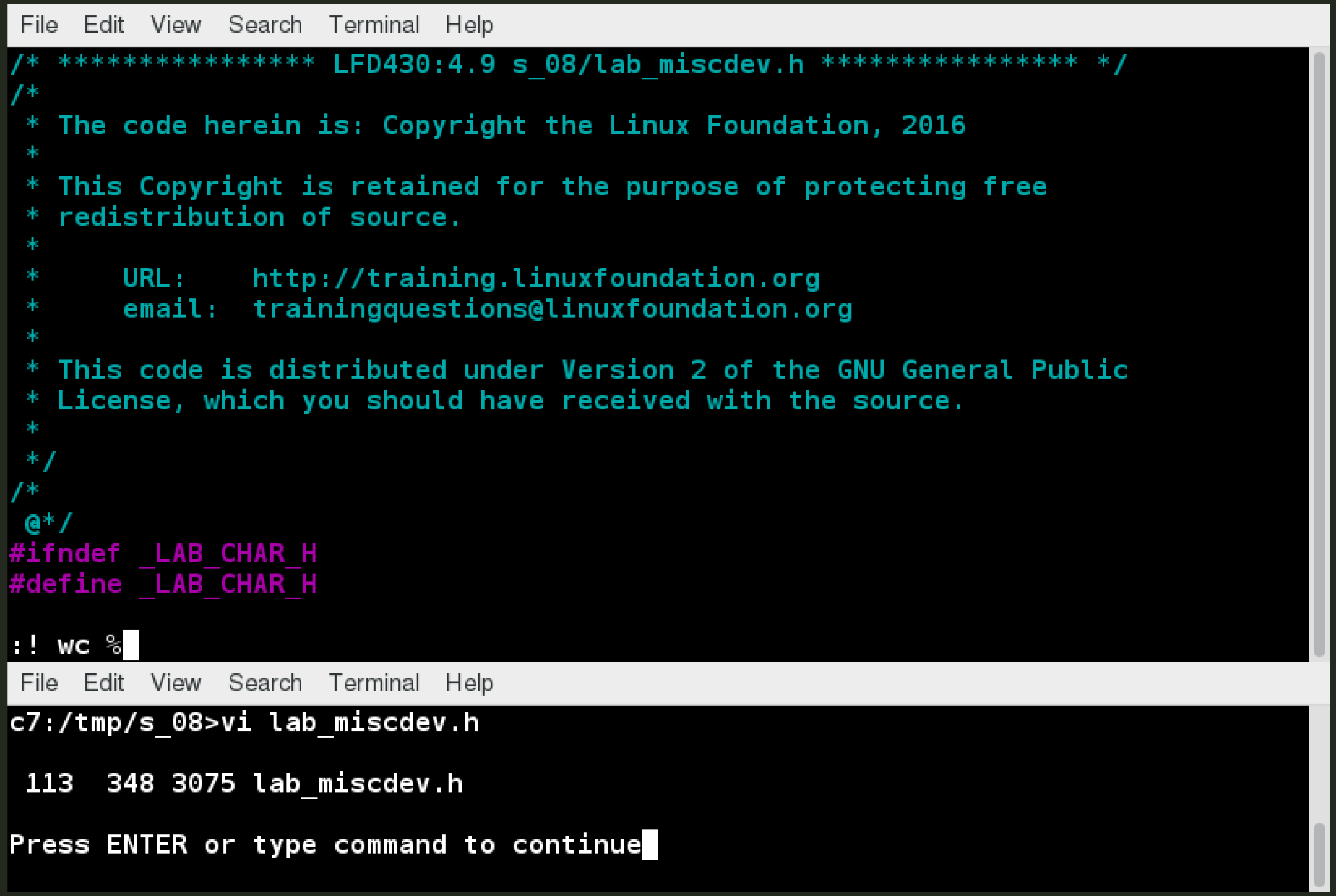
KEY	USAGE
a	Append text after cursor; stop upon **Escape** key
A	Append text at end of current line; stop upon **Escape** key
i	Insert text before cursor; stop upon **Escape** key
I	Insert text at beginning of current line; stop upon **Escape** key
o	Start a new line below current line, insert text there; stop upon **Escape** key
O	Start a new line above current line, insert text there; stop upon **Escape** key
r	Replace character at current position
R	Replace text starting with current position; stop upon **Escape** key
x	Delete character at current position
Nx	Delete N characters, starting at current position
dw	Delete the word at the current position
D	Delete the rest of the current line
dd	Delete the current line
Ndd or **dNd**	Delete N lines
u	Undo the previous operation
yy	Yank (copy) the current line and put it in buffer
Nyy or **yNy**	Yank (copy) N lines and put it in buffer
p	Paste at the current position the yanked line or lines from the buffer

AI STUDIO

Using External Commands in vi

↘ Typing sh command opens an external command shell. When you exit the shell, you will resume your editing session.

↘ Typing ! executes a command from within vi. The command follows the exclamation point. This technique is best suited for non-interactive commands, such as : ! wc %. Typing this will run the wc (word count) command on the file; the character % represents the file currently being edited.



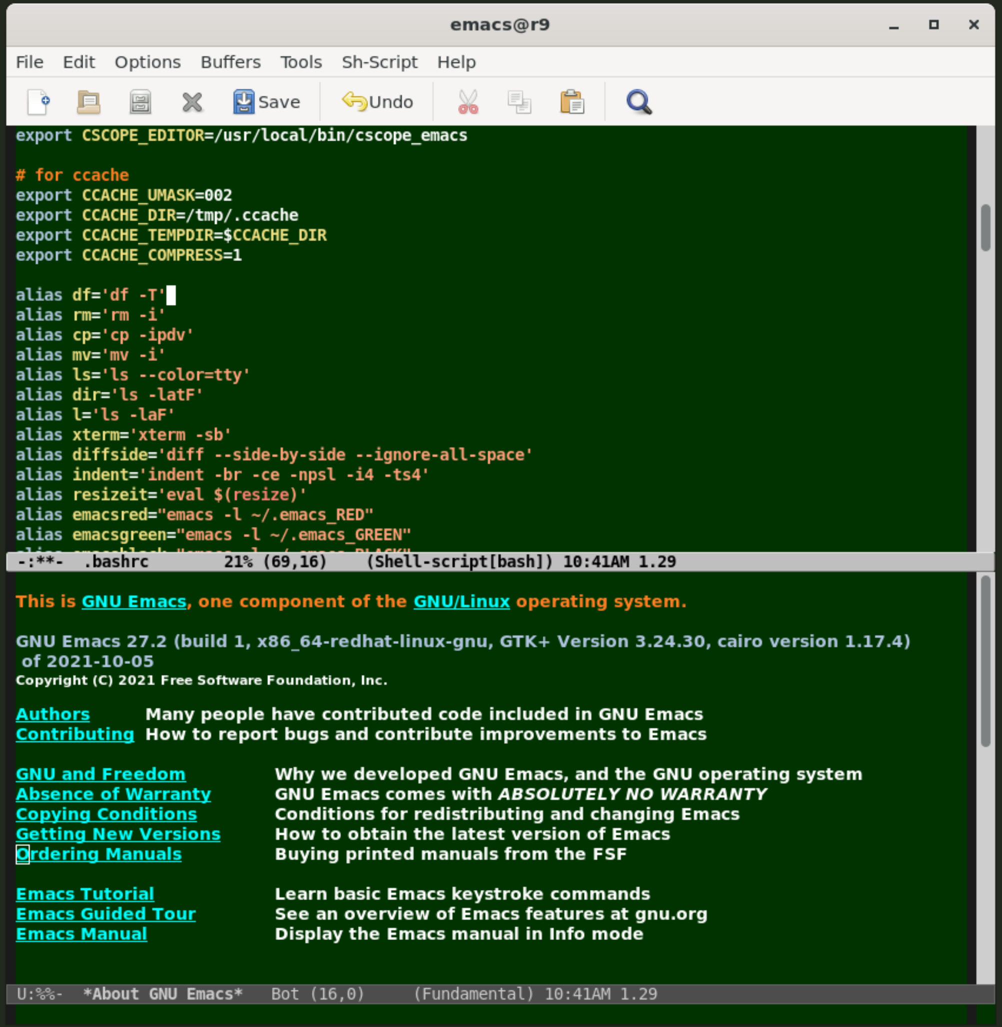
```
/* **************** LFD430:4.9 s_08/lab_miscdev.h **************** */
/*
 * The code herein is: Copyright the Linux Foundation, 2016
 *
 * This Copyright is retained for the purpose of protecting free
 * redistribution of source.
 *
 *     URL:    http://training.linuxfoundation.org
 *     email:  trainingquestions@linuxfoundation.org
 *
 * This code is distributed under Version 2 of the GNU General Public
 * License, which you should have received with the source.
 *
 */
/*
 @*/
#ifndef _LAB_CHAR_H
#define _LAB_CHAR_H

:! wc %
```

```
c7:/tmp/s_08>vi lab_miscdev.h

 113  348 3075 lab_miscdev.h

Press ENTER or type command to continue
```

Introduction to emacs

↘ The emacs editor is a popular competitor for vi. Unlike vi, it does not work with modes. emacs is highly customizable and includes a large number of features. It was initially designed for use on a console, but was soon adapted to work with a GUI as well. emacs has many other capabilities other than simple text editing. For example, it can be used for email, debugging, etc.

↘ Rather than having different modes for command and insert, like vi, emacs uses the CTRL and Meta (Alt or Esc) keys for special commands.



```
export CSCOPE_EDITOR=/usr/local/bin/cscope_emacs

# for ccache
export CCACHE_UMASK=002
export CCACHE_DIR=/tmp/.ccache
export CCACHE_TEMPDIR=$CCACHE_DIR
export CCACHE_COMPRESS=1

alias df='df -T'
alias rm='rm -i'
alias cp='cp -ipdv'
alias mv='mv -i'
alias ls='ls --color=tty'
alias dir='ls -latF'
alias l='ls -laF'
alias xterm='xterm -sb'
alias diffside='diff --side-by-side --ignore-all-space'
alias indent='indent -br -ce -npsl -i4 -ts4'
alias resizeit='eval $(resize)'
alias emacsred="emacs -l ~/.emacs_RED"
alias emacsgreen="emacs -l ~/.emacs_GREEN"
-:**-  .bashrc       21% (69,16)    (Shell-script[bash]) 10:41AM 1.29

This is GNU Emacs, one component of the GNU/Linux operating system.

GNU Emacs 27.2 (build 1, x86_64-redhat-linux-gnu, GTK+ Version 3.24.30, cairo version 1.17.4)
 of 2021-10-05
Copyright (C) 2021 Free Software Foundation, Inc.

Authors      Many people have contributed code included in GNU Emacs
Contributing How to report bugs and contribute improvements to Emacs

GNU and Freedom           Why we developed GNU Emacs, and the GNU operating system
Absence of Warranty       GNU Emacs comes with ABSOLUTELY NO WARRANTY
Copying Conditions        Conditions for redistributing and changing Emacs
Getting New Versions      How to obtain the latest version of Emacs
Ordering Manuals          Buying printed manuals from the FSF

Emacs Tutorial            Learn basic Emacs keystroke commands
Emacs Guided Tour         See an overview of Emacs features at gnu.org
Emacs Manual              Display the Emacs manual in Info mode

U:%%-  *About GNU Emacs*   Bot (16,0)     (Fundamental) 10:41AM 1.29
```

Working with emacs

↘ The table lists some of the most important key combinations that are used when starting, exiting, reading, and writing files in emacs.

↘ The emacs tutorial is a good place to start learning basic commands. It is available any time when in emacs by simply typing CTRL-h (for help) and then the letter t for tutorial.

KEY	USAGE
emacs myfile	Start emacs and edit **myfile**
CTRL-x i	Insert prompted for file at current position
CTRL-x s	Save all files
CTRL-x CTRL-w	Write to the file giving a new name when prompted
CTRL-x CTRL-s	Saves the current file
CTRL-x CTRL-c	Exit after being prompted to save any modified files

AI STUDIO

Changing Cursor Positions in emacs

↘ The table lists some of the keys and key combinations that are used for changing cursor positions in emacs.

KEY	**USAGE**
arrow keys	Use the arrow keys for up, down, left and right
CTRL-n	One line down
CTRL-p	One line up
CTRL-f	One character forward/right
CTRL-b	One character back/left
CTRL-a	Move to beginning of line
CTRL-e	Move to end of line
Meta-f	Move to beginning of next word
Meta-b	Move back to beginning of preceding word
Meta-<	Move to beginning of file
Meta-g-g-n	Move to line n (can also use **'Esc-x Goto-line n'**)
Meta->	Move to end of file
CTRL-v or **Page Down**	Move forward one page
Meta-v or **Page Up**	Move backward one page
CTRL-l	Refresh and center screen

AI STUDIO

Searching & Working in emacs

↘ The table lists the key combinations that are used for searching for text in emacs.

KEY	USAGE
CTRL-s	Search forward for prompted pattern, or for next pattern
CTRL-r	Search backwards for prompted pattern, or for next pattern

KEY	USAGE
CTRL-o	Insert a blank line
CTRL-d	Delete character at current position
CTRL-k	Delete the rest of the current line
CTRL-_	Undo the previous operation
CTRL- (space or **CTRL-@**)	Mark the beginning of the selected region (the end will be at the cursor position)
CTRL-w	Delete the current marked text and write it to the buffer
CTRL-y	Insert at current cursor location whatever was most recently deleted

AI STUDIO

Sample

↘ Assuming myfile does not exist, create it and open it by typing:

```
student:/tmp> vi myfile
```

↘ You now have a mostly blank screen with your cursor on the top line. You are initially in command mode.

↘ Type a to append and put you into insert mode.

↘ Type the sentence:

The quick brown fox jumped over the lazy dog.

If you make a mistake in typing, remember that you can backspace to fix it.

↘ Hit the ESC key. Your cursor should be on the period at the end of the sentence. You are back in command mode at this point.

↘ Type a again, and then hit Enter. You should be on a new line under the one you just typed.

↘ Type the sentence:

Nobody expects the Spanish Inquisition!

Hit Enter key after typing the ! character.

↘ At the beginning of the third line, type the sentence:

This is the third line.

Hit the ESC key. You are now back in command mode.

↘ Now let's just move around in this file. Your cursor should be positioned over the period at the end of the third line you just typed. Hit the h key three times. Hit the k key once.

Which letter is your cursor over? Which word are you in?

Your cursor should be over the S in the word Spanish.

↘ Hit h four times and then hit j. Now which letter is under your cursor? Which word are you in?

Your cursor should be over the r in the word third.

↘ Hit k twice and then l three times. Which letter is your cursor over? In which word?

Your cursor should be over the x in the word fox.

AI STUDIO

Sample

↘ Now hit w eight times. What do you notice? Which letter is your cursor over? Which word is your cursor over?

You should notice that you are now skipping across words. In fact, eight taps on w will take you from the first line to the second line! Your cursor should now be over the e in the word expects on the second line.

↘ Hit k, followed by the $ key. Now hit b twice. Which letter is your cursor over, and in which word?

Your cursor should be over the l in the word lazy.

↘ You should be getting the feel for moving around character-by-character, line-by-line and word-by-word, both forward and backward.

Now hit the 0 (zero) key followed by two w keys and three l keys. Which letter is your cursor over, and in which word are you?

Your cursor should be over the w in the word brown.

↘ Now hit e three times. Which letter is your cursor over, and in which word?

You should be over the d in the word jumped.

↘ Save the file using the :w command. You will use this file in the next exercise. Finally, quit vi by typing the :q command. You can also combine these commands by typing :wq if you prefer.

↘ Hopefully, this practice gets you used to how you can move your cursor around in a file. After a while, these movement keys will become second nature.

↘ The following situation, where you may lose quite a bit of work before it is saved, is likely to come up sooner or later.

↘ Change the permissions on the file you have been editing to read-only, and open it for editing by doing:

```
student:/tmp> chmod 444 myfile
student:/tmp> vi myfile
```

↘ Delete the first line in the file by typing dd. You may see a brief error message, but the line will be deleted.

↘ Now let's try and save the modified file. Type :w. What happens?

You can't save this file, can you? It is read-only! You should have gotten an error message at the bottom line of the screen telling you that the file is readonly.

AI STUDIO

Sample

↘ Perhaps you made lots of changes to this file in a long editing session. Is all your hard work lost? No!

You have a couple of alternatives. The first one is that you can go out to the shell from your editor, change the permissions on the file, exit the shell and be returned to your editing session to save your file.

↘ Do this by typing :sh. This should give you a shell prompt. Now type the command:

```
student:/tmp> chmod 644 myfile
```

and then exit the shell with:

```
student:/tmp> exit
```

You should be returned to your vi editing session. You typically see an error message telling you that the file has changed.

Type O (Shift+o) for OK.

Now type :w! (note the added !). This file should be saved now!

↘ Another thing you could do is to write the contents of the file to a new file. The :w command can take an argument as the file name you want to save the file as. For example, you could type :w new_myfile to save all your changes to the file new_myfile.

↘ It is good to know these techniques because this will likely come up some time while you are working with Linux. Play around with both techniques and become comfortable with both.

AI STUDIO

Summary

↘ Text editors (rather than word processing programs) are used quite often in Linux, for tasks such as creating or modifying system configuration files, writing scripts, developing source code, etc.

↘ nano is an easy-to-use text-based editor that utilizes on-screen prompts.
gedit is a graphical editor, very similar to Notepad in Windows.

↘ The vi editor is available on all Linux systems and is very widely used. Graphical extension versions of vi are widely available as well.

↘ emacs is available on all Linux systems as a popular alternative to vi. emacs can support both a graphical user interface and a text mode interface.

↘ To access the vi tutorial, type vimtutor at a command line window.

↘ To access the emacs tutorial type Ctl-h and then t from within emacs.

↘ vi has three modes: Command, Insert, and Line. emacs has only one, but requires use of special keys, such as Control and Escape.

↘ Both editors use various combinations of keystrokes to accomplish tasks. The learning curve to master these can be long, but once mastered using either editor is extremely efficient.

AI STUDIO

info@aistudio.com.tr

aistudio.com.tr

Thank You!