

1. Ability to work with Vim:

Moving Around in the Vim Editor

 Keys that help you navigate in Vim. Try to avoid using arrow keys; instead, benefit from these keys: up: k, down: j, right: l, left: h

 Move forward to the fourth word in one of the lines, placing the cursor at the beginning of the fourth word: 3w

 Move forward to the fourth word in one of the lines, placing the cursor at the end of the fourth word: 4e, 4b

 Move to the first line of the file: gg

 Move to the last line of the file: G

 Move to the 18th line of the file: 18G


Writing and Deleting

 Go to line 15, go to the end of the line in insert mode, and write your student number: 15G, A

 Go to line 10, select one of the words in the line, and change it to uppercase: 10G, v, w, U

 Return to the beginning of the same line and delete two words. Immediately after deletion, enter writing mode (insert mode): 0, d2wi

 Cut a word you want and paste it at the beginning of line 8: dw, 8GP

 You may have seen in Word software that when insert is activated on the keyboard, you enter overtype mode. This means we can type over existing letters, replacing them one by one. Vim has this capability too; using the middle mouse button, you can do this: R

👉 Go to line 3 and move forward word by word. Then, using the middle mouse button, delete from the current location to the end of the line: `3G, 3w, D`

👉 Copy two lines of your choice. Then go to the end of the file and paste them three times: `2yy, G, 3p`

👉 Copy two words of your choice. Then go to the beginning of the file and paste them. You can use visual mode for this: `v, ewee, y, ggp`

👉 Select several lines and delete the comments. (First select the lines in Visual mode, then go to the beginning of the comments using `#`, and delete them.): `v, j, :, s/^/#/, s/^#//`

👉 Suppose there is a typo in a word and one of its letters needs to be changed. Replace that letter without entering Insert mode: `r<char>`

🔵 Searching and Running Commands

👉 Search for a word in the text and move between the results. For example, in a config file like `ssh`, search for the word `key` or `host`: `/word, n` for down and `N` for up

👉 Search for the same word again, in a way that it doesn't consider uppercase and lowercase differences: `/\word`

👉 Enter a command to replace the desired word written in the entire file with `new_word`. This replacement must be case-insensitive, and for each change, your confirmation must be received first: `:%s/old_word/new_word/gci`


👉 Add a setting to Vim so that the search results are displayed highlighted: `vim ~/.vimrc, set hlsearch`

👉 Run a Linux command from within Vim without exiting. For example, to temporarily edit a config file, you need a tool that shows


your IP, and you want to run this in the Vim editor. Run the following command: `:!command`


Saving, Comparing, and Working with tab and window

 Make changes in a new file and save them: `:w`

 Copy six consecutive lines from a file and then paste them into a new Vim tab. Save the new file with a new name newtab: `6V, y, :tabnew, p, :w newtab`

 Create a new window in the new tab and copy and paste the first fifteen lines into it: `15V, y, :sp, Ctrl+w w, p`

 In the new window, delete a few words continuously and change a few words. Then save the file with the name for-diff and exit Vim: `Ctrl+w w, dw, cw, :wq for-diff`

 Compare the original file and the file for-diff using the command `vimdiff`, and move between the different lines of the two files using the commands `:diffput` and `:diffget`: `vimdiff ssh_config for-diff`

2. Defining custom shortcut keys (Key mapping)

a) `nnoremap <leader>w :w<CR>`

b) `nnoremap <leader>v :vsp<CR>`

`nnoremap <leader>h :sp<CR>`

Configuring editor:

Basic Line Numbers: set number

Relative Line Numbers: set relativenumber

Case-Insensitive: set ignorecase

Set Tab to 4 Spaces: set tabstop=4, set softtabstop=4

Enable Syntax Highlighting: syntax enable

Set a Color Scheme: colorscheme desert

3. Add plugin to Vim

a) `curl -fLo ~/.vim/autoload/plug.vim --create-dirs \`
<https://raw.githubusercontent.com/junegunn/vim-plug/master/plug.vim>

`vim ~/.vimrc`

Add this:

```
call plug#begin('~/.vim/plugged')
```

```
Plug 'preservim/nerdtree'
```

```
call plug#end()
```

:PlugInstall

Open NERDTree with: :NERDTreeToggle

Close NERDTree with: :NERDTreeClose

b) nnooremap <leader>n :NERDTreeToggle<CR>



The screenshot shows a Vim editor window with a dark background. The left pane displays the file explorer, showing the current directory as /home/erfan/Desktop/. The right pane shows the .vimrc file, which contains the following configuration:

```
1 # ssh_config(5) for more information. This file provides defaults for
2 # users, and the values can be changed in per-user configuration files
3 # or on the command line.
4
5 # Configuration data is parsed as follows:
6 # 1. command line options
7 # 2. user-specific file
8 # 3. system-wide file
9 # Any configuration value is only changed the first time it is set.
10 # Thus, host-specific definitions should be at the beginning of the
11 # configuration file, and defaults at the end.
12
13 # Site-wide defaults for some commonly used op
14 # ssh_config(5) man page.
15
16 Include /etc/ssh/ssh_config.d/*.conf
17
18 nnooremap <leader>h :sp<CR>
19
20 nnooremap <C-n> :NERDTreeToggle<CR>
21
22 nnooremap <leader>w :w<CR>
23
24 set number
25 set relativenumber
26 set ignorecase
27 set tabstop=4
28 set softtabstop=4
29 syntax enable
30 colorscheme desert
31 call plug#begin('~/.vim/plugged')
32 Plug 'preservim/nerdtree'
33 call plug#end()
34
35
```

c) I added vim-airline to the Plugins. vim-airline is a lightweight status/tabline plugin for Vim that provides a clean, informative, and customizable bar at the bottom of your editor window.

Add *Plug 'vim-airline/vim-airline'* to *.vimrc* inside the *call plug#begin()* and *call plug#end()* block. Then run *:PlugInstall*

```
1  set hlsearch
2  nnoremap <leader>v :vsp<CR>
3  nnoremap <leader>h :sp<CR>
4  nnoremap <C-n> :NERDTreeToggle<CR>
5  nnoremap <leader>w :w<CR>
6  set number
7  set relativenumber
8  set ignorecase
9  set tabstop=4
10 set softtabstop=4
11 syntax enable
12 colorscheme desert
13 call plug#begin('~/.vim/plugged')
14 Plug 'preservim/nerdtree'
15 Plug 'vim-airline/vim-airline'
16 call plug#end()
17
18
19
20
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54
55
56
57
58
59
60
61
62
63
64
65
66
67
68
69
70
71
72
73
74
75
76
77
78
79
80
81
82
83
84
85
86
87
88
89
90
91
92
93
94
95
96
97
98
99
100
```

NORMAL ~/.vimrc

vim utf-8[unix] 5% ln: 1/18

4. Using micros in Vim

```
:%s/([^\,]*)\,([^\,]*)\.([^\,]*)/Poet: \1 | Years: \2 | Famous Work: \3/
```

qt

```
:s/([^\,]*)\,([^\,]*)\,([^\,]*)/Poet: \1 | Years: \2 | Famous Work: \3/<Enter>
```

```
:%norm! @t
```


۳) ویرایشگری مثل Nano ممکن است یادگیری ساده‌تری نسبت به Vim داشته باشد، اما در عوض قدرت و انعطاف‌پذیری Vim بسیار بیشتر است. افراد حرفه‌ای و مدیران سیستم (Sysadmin) معمولاً Vim را انتخاب می‌کنند چون می‌توانند کارها را سریع‌تر، دقیق‌تر و بدون استفاده از موس انجام دهند. Nano بیشتر برای ویرایش‌های سریع و ساده مناسب است.

۴) استفاده از ماکرو در Vim هنگام ویرایش متن باعث صرفه‌جویی در زمان و انرژی می‌شود. می‌توانیم یکسری اقدامات تکراری را یکبار ضبط کرده و سپس بارها در نقاط مختلف فایل اجرا کنیم. این موضوع در پروژه‌های بزرگ یا هنگام فرمت‌بندی داده‌ها بسیار مفید است.

۵) یکی از مشکلات رایج در استفاده از ماکروها، عدم تطابق شرایط در حین Replaying با شرایط ضبط شده است. مثلاً اگر موقع اجرای ماکرو یک خط خالی یا متفاوت نسبت به حالت ضبط وجود داشته باشد، ممکن است ماکرو درست کار نکند. بنابراین باید دقت کنید ماکرو در شرایط مشابه اجرا شود. در این بخش به چنین مشکلی بر نخوردم.

۶) Modeless مثل Notepad همیشه در حالت نوشتن هستند، اما Modal مثل Vim بین حالت‌های مختلفی مثل Normal، Insert، Visual و غیره جابجا می‌شود. Vim در دسته Modal قرار می‌گیرد که این ویژگی باعث می‌شود با کلیدهای خاصی مثل c, y, d کارهای پیچیده‌ای انجام دهید بدون اینکه موس استفاده کنیم.

۷) در Vim حالت‌های مختلفی مثل Normal، Insert، Visual و Command وجود دارد. با تصویر نشان داده شده بهتر می‌توان فهمید که چطور از یک حالت به دیگری می‌رویم. مثلاً با زدن Esc به حالت Normal برمی‌گردیم، یا با زدن : وارد Command mode می‌شویم. درک این حالت‌ها کلید تسلط بر Vim است.