
بسم الله الرحمن الرحيم

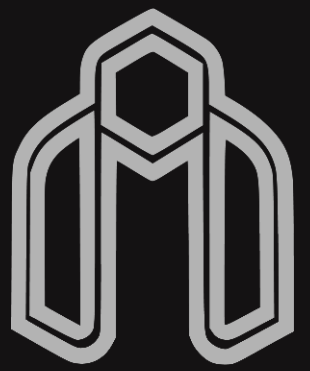
LINUX COMMANDS

Feel free

Description of some Linux commands – Part VI
OS Laboratory – Exercise 6

Collected and Written by Mostafa Fazli

30 November 2021



vim

nano

sublime

gcc

g++

echo

return code

git

vim

vim, which stands for "Vi Improved", is a text editor. It can be used for editing any kind of text and is especially suited for editing computer programs.

There are a lot of enhancements above **Vi**: multi level undo, multiple windows and buffers, syntax highlighting, command line editing, file name completion, a complete help system, visual selection, and others.

It has many modes like INSERT mode, REPLACE mode and …

Structure:

vim [OPTION[[FILE_NAME]

- You can install Vim with : sudo apt install vim

vim

OPTIONS:

(*NO OPTION) file_name => Make new file with file_name

- (dash) => A single dash specifies that the file to edit is to be read from standard input.

vim

When you are in Vim Text editor Environment:

You can use this keys in normal mode for move around:

H => left

J => down

K => up

I => right

vim

When you are in Vim Text editor Environment:

Editing commands:

-d => start deleting

-d => delete a line

-dw => delete a word

Ctrl + r => redo

vim

When you are in Vim Text editor Environment:

Searching:

/ => for search a string

n => go to next data

N => go to previous data

vim

When you are in Vim Text editor Environment:

Copy and Paste:

-yy => copy a line

-yw => copy a word

-p => paste

vim

When you are in Vim Text editor Environment:

Save and exit:

:wq => save and exit

:q! => exit without save

vim

When you are in Vim Text editor Environment:

You can change font with this command:

:set guifont = courier

Change color with:

:colorscheme <tab>

Text Manipulation

i	insert before cursor
I	insert at start of line
a	insert after cursor
A	insert at end of line
o	add new line below cursor
O	add new line above cursor
ea	insert at end of line
Esc	exit insert mode

r	replace single character
cc	replace line
cw	replace to end of word
c$	replace to end of line
s	substitute character
S	substitute line
u	undo
Ctrl	redo

Visual Mode

v	enter visual mode
V	enter linewise visual mode
Ctrl	start visual block mode
>	shift text left
<	shift text right
>>	shift left by shiftwidth
<<	shift right by shiftwidth
==	auto-indent line

~	change case
Esc	exit visual mode

k	move cursor up
H	jump to first line in document

h	move cursor left
b	move to start of previous word
B	move to start of previous word (inc. puctuation)
0	move to start of line

l	move cursor right
w	move to start of next word
W	move to start of next word (inc. punctuation)
$	move to end of line

j	move cursor down
G	jump to last line in document

Text Manipulation Cont.

y	yank/copy
yy	yank a line
yw	yank a word
y$	yank to end of line
p	paste after cursor
P	paste before cursor
dd	delete/cut a line
dw	delete a word

D	delete to end of line
x	delete character
/string	search for “string”

Save & Exit

ZZ	save and quit
ZQ	quit without saving
:w	write/save
:q	quit (fails if there are changes)
:wq	write and quit
:x	write and quit
:q!	force quit without saving
:qa	quit all vim buffers

mostafafazli@pop-os:~/Documents/OSLaboratory/MainTest | 93x43 | pts/0

```
┌──(13:52:19)──> ls                                                    ──(Tue,Nov30)─┐
MainTest  Test1
┌ ~/Documents/OSLaboratory ──────────────────────────── mostafafazli pop-os:pts/0 ┐
└──(13:52:19)──> cd MainTest                                         ──(Tue,Nov30)─┘
┌ ~/Documents/OSLaboratory/MainTest ─────────────────── mostafafazli pop-os:pts/0 ┐
└──(13:52:24)──> ls -l                                               ──(Tue,Nov30)─┘
total 0
┌ ~/Documents/OSLaboratory/MainTest ─────────────────── mostafafazli pop-os:pts/0 ┐
└──(13:52:26)──> vim VimTest.txt                                     ──(Tue,Nov30)─┘
┌ ~/Documents/OSLaboratory/MainTest ─────────────────── mostafafazli pop-os:pts/0 ┐
└──(13:53:52)──> ls -l                                               ──(Tue,Nov30)─┘
total 4
-rw-rw-r-- 1 mostafafazli mostafafazli 119 Nov 30 13:53 VimTest.txt
┌ ~/Documents/OSLaboratory/MainTest ─────────────────── mostafafazli pop-os:pts/0 ┐
└──(13:53:55)──> cat VimTest.txt                                     ──(Tue,Nov30)─┘
Hello,
This is a test for Os laboratory Exercise
I'm Mostafa Fazli
This text written with Vim TextEditor.
End of text
┌ ~/Documents/OSLaboratory/MainTest ─────────────────── mostafafazli pop-os:pts/0 ┐
└──(13:54:11)──>                                                     ──(Tue,Nov30)─┘
```

OSLaboratory MainTest

- Recent
- Starred
- Home
- Desktop
- Documents
- Downloads
- Music
- Pictures
- Videos
- Trash
- Other Locations

VimTest.txt

Open VimTest.txt ~/Documents/OSLabor... Save

```
1 Hello,
2 This is a test for Os laboratory Exercise
3 I'm Mostafa Fazli
4 This text written with Vim TextEditor.
5 End of text
6
```

Plain Text Tab Width: 8 Ln 6, Col 1 INS

nano

GNU nano is an easy to use command line text editor for Unix and Linux operating systems. It includes all the basic functionality you'd expect from a regular text editor, like syntax highlighting, multiple buffers, search and replace with regular expression support, spellchecking, UTF-8 encoding, and more.

- You can install nano with : sudo apt install nano

Structure:

nano [OPTION] [FILE_NAME]

nano

OPTIONS:

(*NO OPTION) file_name => Make new file with file_name

nano

Crtl + x => exit file, then you determine save file or not, if you opened for first time you should set a name for file.

Ctrl + \ => for replace a text

Ctrl + w => for search a text

Ctrl + k => cut text

Alt + 6 => copy text

Ctrl + U => paste text

Alt + E => redo

nano

Ctrl+F : move forward one character

Ctrl+B : move back one character

Ctrl+Space : move forward one word

Alt+Space : move back one word

Ctrl+P : move to the previous line

Ctrl+N : move to the next line

Ctrl+V : move to the next page

Ctrl+Y : move to the previous page

Ctrl+A : move to the beginning of the line

Ctrl+E :move to the end of the line

nano NanoTest.txt

```
  GNU nano 4.8                    NanoTest.txt
Hello again
This is a test for Os laboratory exercise too
Im Mostafa Fazli again
This Text written with Nano TextEditor.
and end of text is here



                    [ Read 6 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 Paste Text ^T To Spell ^_ Go To Line
```

Home Documents OSLaboratory MainTest

- Recent
- Starred
- Home
- Desktop
- Documents
- Downloads
- Music
- Pictures
- Videos
- Trash
- Other Locations


NanoTest.txt


VimTest.txt

mostafafazli@pop-os:~/Documents/OSLaboratory/MainTest | 93x43 | pts/0

```
┌─ ~/Documents/OSLaboratory/MainTest ──────────── mostafafazli pop-os:pts/0 ─┐
└─(13:54:31)──> ls -l                                          ──(Tue,Nov30)─┘
total 4
-rw-rw-r-- 1 mostafafazli mostafafazli 119 Nov 30 13:53 VimTest.txt
┌─ ~/Documents/OSLaboratory/MainTest ──────────── mostafafazli pop-os:pts/0 ─┐
└─(13:54:34)──> nano NanoTest.txt                              ──(Tue,Nov30)─┘
┌─ ~/Documents/OSLaboratory/MainTest ──────────── mostafafazli pop-os:pts/0 ─┐
└─(13:56:21)──> ls -l                                          ──(Tue,Nov30)─┘
total 8
-rw-rw-r-- 1 mostafafazli mostafafazli 146 Nov 30 13:56 NanoTest.txt
-rw-rw-r-- 1 mostafafazli mostafafazli 119 Nov 30 13:53 VimTest.txt
┌─ ~/Documents/OSLaboratory/MainTest ──────────── mostafafazli pop-os:pts/0 ─┐
└─(13:56:27)──> cat NanoTest.txt                               ──(Tue,Nov30)─┘
Hello again
This is a test for Os laboratory exercise too
Im Mostafa Fazli again
This Text written with Nano TextEditor.
and end of text is here

┌─ ~/Documents/OSLaboratory/MainTest ──────────── mostafafazli pop-os:pts/0 ─┐
└─(13:56:32)──>                                                ──(Tue,Nov30)─┘
```

Home Documents OSLaboratory MainTest

- Recent
- Starred
- Home
- Desktop
- Documents
- Downloads
- Music
- Pictures
- Videos
- Trash
- Other Locations


NanoTest.txt


VimTest.txt

sublime

Sublime Text is a commercial source code editor. It natively supports many programming languages and markup languages. Users can expand its functionality with plugins, typically community-built and maintained under free-software licenses. To facilitate plugins, Sublime Text features a Python API.

sublime

Features:

The following is a list of features of Sublime Text

•"Goto Anything," quick navigation to files, symbols, or lines

•"Command palette" uses adaptive matching for quick keyboard invocation of arbitrary commands

•Python-based plugin API

•Project-specific preferences

•Extensive customizability via JSON settings files, including project-specific and platform-specific settings

•Cross-platform (Windows, macOS, and Linux) and Supportive Plugins for cross-platform

gcc

GCC stands for GNU Compiler Collection,

It is an optimizing compiler produced by the GNU Project supporting various programming languages, hardware architectures and operating systems.

It is mainly used to compile the C and C++ programs.

It takes the name of the source program as a necessary argument;
rest arguments are optional such as debugging, warning, object file,
and linking libraries.

GCC is a core component of the GNU toolchain.

g++

g++ command is a GNU c++ compiler invocation command, which is used for preprocessing, compilation, assembly and linking of source code to generate an executable file. The different “options” of g++ command allow us to stop this process at the intermediate stage.

Structure:

g++ [OPTION] [FILE_NAME]

--version => check g++ compiler version

File_name => compile a file to generate executable file

* You can open executable file with “./” like : ./a.out

```
string printZero(){

    return "0";

}

string printNonZero(){


    string str = "1";
    return str;

}

int main(){

    int n;
    cin >> n;

    if(n == 0) {
        cout << "You Entered 0, now We return " + printZero();
        return 0;
    }
    else{
        cout << "You dosn't Entered 0, now We return anything else of 0 like " + printNonZero();
        return 1;
    }

}
```

```
~/Documents/OSLaboratory/MainTest mostafafazli pop-os:pts/0
(14:10:53)—> g++ Program.cpp (Tue,Nov30)
~/Documents/OSLaboratory/MainTest mostafafazli pop-os:pts/0
(14:10:58)—> ls -l (Tue,Nov30)
total 32
-rwxrwxr-x 1 mostafafazli mostafafazli 18464 Nov 30 14:10 a.out
-rw-rw-r-- 1 mostafafazli mostafafazli   146 Nov 30 13:56 NanoTest.txt
-rw-rw-r-- 1 mostafafazli mostafafazli   464 Nov 30 14:10 Program.cpp
-rw-rw-r-- 1 mostafafazli mostafafazli   119 Nov 30 13:53 VimTest.txt
~/Documents/OSLaboratory/MainTest mostafafazli pop-os:pts/0
(14:11:01)—> ./a.out (Tue,Nov30)
0
You Entered 0, now We return 0%
~/Documents/OSLaboratory/MainTest mostafafazli pop-os:pts/0
(14:11:10)—> (Tue,Nov30)
```

echo

This command used for print a text.

Sometimes this command combine with other commands (pipeline) , for example:
echo “deb https://download.telegram.org/apt/....” | sudo tee /etc/apt/sources.list.d/telegram.list

Sometimes just for print a text like:
echo Hello World!!!
And its print:
Hello World!!!

echo

One of another use of echo, change a text, like :

Structure:
echo [OPTION(s)] [STRING(s)]

(* no option) => print string in ormal mode
$ => for declare a variable you should to use $
\e => for remove \
\n => for print each \ in one line
\t => add tab after each \
\v => print each line in vertical tab (I like this mode)
...
* It has a lot of options, these options is more importand.

```
mostafafazli pop-os:p
4)—> echo Mostafa Fazli                                    —(Tue,M
zli
mostafafazli pop-os:p
8)—> echo -e "Mostafa \nFazli"                             —(Tue,M


mostafafazli pop-os:p
1)—> echo -e "Mostafa \tFazli"                             —(Tue,M
     Fazli
mostafafazli pop-os:p
6)—> echo -e "\aMostafa Fazli"                             —(Tue,M
zli
mostafafazli pop-os:p
8)—>                                                       —(Tue,M
```

return code

When the Batch Client exits it returns a result code to the calling program based on the exit condition of the script. There are two possible scenarios based on the StopOnError setting.

StopOnError = False (-S0)

If StopOnError is false the return code indicates general success or failure.

0 = success (no errors)

-1 = failure (one or more errors occurred)

StopOnError = True (-S1)

return code

On the other words return code determine running program is Successful or not,

If successful and without any errors return 0 and if it isn’t return anything else.

Command	Command Code	Class	Class Code	Return Code
Success	N/A	N/A	N/A	0
General Error	N/A	N/A	N/A	-1
Validation Error	N/A	N/A	N/A	1
Parse Error	N/A	N/A	N/A	100
Command Line Error	N/A	N/A	N/A	4
Copy	15	Application	1	1501
Copy	15	Dimension	2	1502
Create	1	Application	1	101
Create	1	Dimension	2	102
Create	1	Member	3	103
Create	1	Association	10	110
Debug	21	N/A	N/A	2100
Delete	2	Application	1	201
Delete	2	Dimension	2	202
Delete	2	Member	3	203
Delete	2	Association	10	210
Detach	16	Dimension	2	1602
Exclude	3	Member	3	303
Execute	4	DataSynchronization	4	404
Execute	4	Deploy	5	405
Execute	4	DimensionSynchronization	6	406
Execute	4	Import	7	407

git

Git is software for tracking changes in any set of files, usually used for coordinating work among programmers collaboratively developing source code during software development. Its goals include speed, data integrity, and support for distributed, non-linear workflows (thousands of parallel branches running on different systems).

* This is best definition of Git that I can say, it taken from Wikipedia.

History of git:

Git is a free and open-source version control system, originally created by Linus Torvalds in 2005. Unlike older centralized version control systems such as SVN and CVS, Git is distributed: every developer has the full history of their code repository locally. This makes the initial clone of the repository slower, but subsequent operations such as commit, blame, diff, merge, and log dramatically faster.

git

Commands:

git config => This command sets the author name and email address respectively to be used with your commits.

git init => This command is used to start a new repository.

git clone => This command is used to obtain a repository from an existing URL.

git add => This command adds a file to the staging area.

git commit => This command records or snapshots the file permanently in the version history.

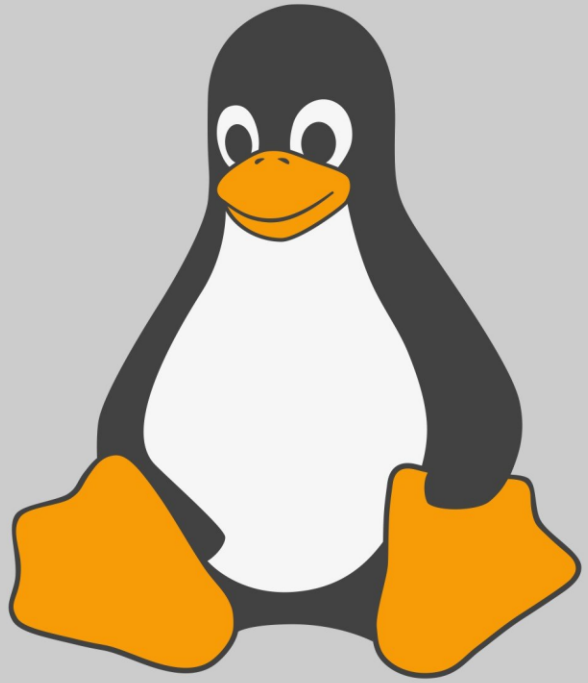
git diff => This command shows the file differences which are not yet staged.

git status => This command lists all the files that have to be committed.

Git rm => This command deletes the file from your working directory and stages the deletion.

Git log => This command is used to list the version history for the current branch.

```
└─(14:14:48)──> git                                                    ──(Tue,Nov30)─┘
usage: git [--version] [--help] [-C <path>] [-c <name>=<value>]
           [--exec-path[=<path>]] [--html-path] [--man-path] [--info-path]
           [-p | --paginate | -P | --no-pager] [--no-replace-objects] [--bare]
           [--git-dir=<path>] [--work-tree=<path>] [--namespace=<name>]
           <command> [<args>]

These are common Git commands used in various situations:

start a working area (see also: git help tutorial)
   clone             Clone a repository into a new directory
   init              Create an empty Git repository or reinitialize an existing one

work on the current change (see also: git help everyday)
   add               Add file contents to the index
   mv                Move or rename a file, a directory, or a symlink
   restore           Restore working tree files
   rm                Remove files from the working tree and from the index
   sparse-checkout   Initialize and modify the sparse-checkout

examine the history and state (see also: git help revisions)
   bisect            Use binary search to find the commit that introduced a bug
   diff              Show changes between commits, commit and working tree, etc
   grep              Print lines matching a pattern
   log               Show commit logs
   show              Show various types of objects
   status            Show the working tree status

grow, mark and tweak your common history
   branch            List, create, or delete branches
   commit            Record changes to the repository
   merge             Join two or more development histories together
   rebase            Reapply commits on top of another base tip
   reset             Reset current HEAD to the specified state
   switch            Switch branches
   tag               Create, list, delete or verify a tag object signed with GPG

collaborate (see also: git help workflows)
   fetch             Download objects and refs from another repository
   pull              Fetch from and integrate with another repository or a local branch
   push              Update remote refs along with associated objects

'git help -a' and 'git help -g' list available subcommands and some
concept guides. See 'git help <command>' or 'git help <concept>'
to read about a specific subcommand or concept.
See 'git help git' for an overview of the system.
┌─ ~ ──────────────────────────────────────────────── mostafafazli pop-os:pts/0 ─┐
└─(14:14:58)──>                                             1 ↵ ──(Tue,Nov30)─┘
```



LINUX