Bash command line and Git

Sofiullah Iqbal Kiron

31 March, 2021

Last compiled: October 23, 2021

If git is already installed in your system, then bash command line interface are installed also. Type and run command from it, the command might be Linux, UNIX or standard git command. All of the git command starts with keyword `git`.
Simple bash commands:

Bash Commands

1. To get help or print manual list of any command just type `command --help`

2. Sometimes we want to create a new file or there may be times when the requirement is to change the timestamps of a file. The `touch [option]...FILE...` command is primarily used to change file timestamps, but if the file(whose name is passed as an argument with extension) doesn't exist, the the command will create it. Ex. `touch newText.txt` will create a new text file at working directory by that specified name if not already exist.

3. To remove a directory and all its contents, including sub-directories and files (cmd command): `rm -r "directory name"` r stands for recursive. That means recursively remove all.

4. Powershell version: Run as administrator, type `$PSVersionTable`, hit enter.

5. Linux/UNIX command to write something on a file: `echo message > fileNameWithExtension` Ex: `echo Hello > file.txt`

6. Directory:

   (a) Make: `mkdir dirName`

   (b) Change working directory: `cd dirNameWithFullPath`

   (c) Path of working directory: `pwd`

7. `Start`:

   (a) `explorer`

   (b) `texmaker`

   (c) `msedge` or `msedge <link>`

   (d) `chrome` or `chrome <link>`

   (e) `brackets`

   (f) `opera` or `opera <link>`

   (g) `cmd`

   (h) `powershell`

8. List:

   (a) Simple: `ls / dir`

   (b) Hidden included: `ls/dir -a`

   (c) Hidden included with details: `ls/dir -all`

9. Let's some fun:

   (a) `factor n` will print prime factors of `n`. It is a build-in program like we say.

10. Compile TeX files (must have a compiler like MiKTeX or TeXLive pre-installed on your system):
    (a) Produce PDF from TeX: `pdflatex [filename].tex`
    (b) Produce DVI from TeX: `latex [filename].tex`
    (c) DVI to PDF: `dvipdfm [filename].dvi`
11. Open current directory on VS code: `code .`
12. Stop execution of a running command: `ctrl + c`
13. If we wanna see all the commands(that we typed and hit enter, not matter right or wrong the command was) as a list: `history`
14. Clear the history: `history -c`
15. Clear console: `clear`
16. Exit from bash terminal: `logout`

Git commands

1. Check git version: `git --version`
2. Initialize a new empty local repository: `git init` :- a hidden folder will be created with name ".git"
3. Adding/Staging(give access to git tracker) files:
    (a) Specified: `git add fileName`
    (b) Multiple: `git add file1 file2 file3`, space separated.
    (c) Specified by extension: `git add *.ext`
    (d) All: `git add .`
4. Staging logs:
    (a) Full log: `git log`, press 'q' to quit from long log list. Each log come up with a unique hexadecimal ID(String with 40 characters maybe). Each commit contains a complete snapshoot of working directory. log is the history the snapshoots of a repository.

(b) Press space for move to the next page of log.

(c) One lined: `git log --oneline`

(d) Last N commits by a non negative integer: `git log -N`, N means non-negative integer as, 2, 6. eg. `git commit -3` will show last 3 log.

(e) In reverse order, full: `git log --reverse`.

(f) In reverse order, one lined: `git log --oneline --reverse`.

(g) Show changed files in each commit: `git log --stat` or `git log --oneline --stat`

(h) See actual changes in each commit: `git log --patch` or `git log --oneline --patch`

(i) Show commit changes: `git show commitID` or, `git show HEAD n`, here HEAD is the last commit we did and n is an integer indicating that go n steps back from HEAD and then show the commit changes.

(j) Filter logs to be shown by

i. Author: `git log --author="author_name"`

ii. Date

iii. Before with this date: `git log --before="yyyy-mm-dd"`

iv. After with this date: `git log --after="yyyy-mm-dd"`

v. Indication: `git log --before/after="yesterday/one week ago/two week ago/one month ago/two month ago"`

vi. the commit messages those has the string/substring "WORD". It is case sensitive. : `git log --grep="WORD"`

5. File list in staging area: `git ls-files`

6. File list with tree: `git ls-tree commitID` or, `git ls-tree HEAD n`

7. Status:

(a) Full status: `git status`

(b) Short status: `git status -s`, I think, `-s` stands for "short".

8. Configuration:

(a) Full: `git config --list`

(b) Specified by key: `git config <key>` e.g. To get user name that already been configured, git config user.name

(c) `git config --global user.name "My Name"`

(d) `git config --email myEmail`

9. Get a full copy of an existing repository: `git clone <url>`

10. Set default editor for git

11. Open the dot git folder: `start .git`

12. Remove tracking access from git: delete the `.git` hidden directory.

13. Update git: `git update-git-for-windows`

14. Every git commit contains a distinct ID, Message, Date/Time, Author, Complete snapshot of project.

15. Remote:

    (a) Remote repository is a repository exits on server/online.

    (b) Add: `git remote add <shortname> <url>`

    (c) Lists all the remotes with server link details: `git remote -v`

    (d) Inspect: If you want to see more information about a particular remote: `git remote show <remoteShortName>`

    (e) Rename: `git remote rename <oldShortName> <newShortName>`

    (f) Remove: `git remote rm <shortname>` or `git remote remove <shortname>`

    (g) To see remote servers you have configured, you can run the `git remote` command. It lists the shortnames of each remote handle you've specified. If you've cloned your repository, you should at least see "origin" that is the default name Git gives to the server you cloned from. Can use `-v` option to shows Related URLs that git has stored for the shortname.

16. **Have Bug**, After adding a remote to a repository, we can fetch by the shortname: `git fetch <shortname>`

Kali Linux

1. Kali Linux is a Linux distro based on debian specially made for hackers cause kali has a lot of hacking tools and penetration testing tools preinstalled and preconfigured in it.
2. We use python for ethical hacking and will make many many tools with this. Automate.
3. Know who is the user: whoami
4. Internet configuration: ifconfig
5. Three must have skill: gathering information,
6. `cat` *(concatenate)* command
   (a) Show all the contents of given filename: `cat file_name`
   (b) Show contents of multiple files: `cat file1 file2`
   (c) Show contents preceding with line numbers: `cat -n file_name`
   (d) Create a file: `cat > file_name.ext`
   (e) Copy content of file1 to file2: `cat file1 > file2`
   (f) : `cat`
   (g) : `cat`

0.1 Password Cracking

Passwords are stored inside a server in a hash(crazy, massive numbers and letters) form, not in original plain text from. Options: Rainbow table, dictionary attack, brute force method, hashcat,
For brute force attack we use hydra that are preinstalled on Kali Linux.

Sherlock

Sherlock is a CLI based python program that can find all social media accounts by a username that provided as a parameter. The name taken from famous detectives name: "Sherlock Homes" (I like the guy, yeah!). Source link on github:

Blackeye: The ultimate phishing tool for free

Blackeye is a great phishing tool for hackers. Here is the complete instruction: Before this: Create and connect your **ngrok** account through terminal to get localhost (one for free). Or if it already exists on your system then kill the previous process by this command: `kill -9 PID`[1] (You need to find out that wheres the process is running, may be it on the sites folder). To get the PID run the command: `top` and find PID of **ngrok**. `ctrl + c` to get out from the process.

1. Get blackeye: `git clone https://github.com/An0nUD4Y/blackeye.git`
2. Change directory to blackeye: `cd blackeye`
3. Run blackeye.sh:
   (a) `bash blackeye.sh`[2]
   (b) `chmod +x blackeye.sh` → `./blackeye.sh`
4. Choose an option
5. While the process is running, do this job: open two terminal, one on blackeye, second on that sites folder that you wanna phish. Run this command on first terminal: `./ngrok http 8080`, before this, run this command on second terminal: `php -S localhost:8080` .
6. Now at the first terminal, the phishing link is ready to go...
7. Wait for victim to response.
8. After loging into the phishing site. It will take the victim to original page.

Basically, blackeye tool is owned by **thelinuxchoice**.
Cautions:

1. Become anonymous, zsecurity has a tutorial on youtube for this.
2. Change the mac address:

[1]Process ID
[2]better choice

(a) `ifconfig eth0 down` to disable ethernet connection eth0

(b)

(c) `ifconfig eth0 up`

3. Get connected with a VPN.

4. Attack with the help of Social Engineering as: `Hey, it's an important message for you. Please check it out right now.` Or, `Check your facebook security and make it strong.`, etc. Also can use my various corporate g-mail accounts for this. Create a g-mail from non-native country with the same name as company or site like, *facebooksecurityhub3032@gmail.com*

5. Create another git and github account for hacking.

6. Modifying the phishing *website.html* from blackeye/sites directory would be good to phishing attack. Learn **HTML**, **CSS** and **PHP** well. Or insect the code from original page with the help of browser insect option.

7.

This tutorial pdf is for educational purposes only. Do not hack anyone without permission.

Some issues: *how to fix blackeye tool for phishing website link is not showing issue, how to fix err_ngrok_108 issue, what is local host and domain, what is dns server, what is ubuntu server and why should we use it, what is dns poisoning.*

Mail: I will use for educational hacking

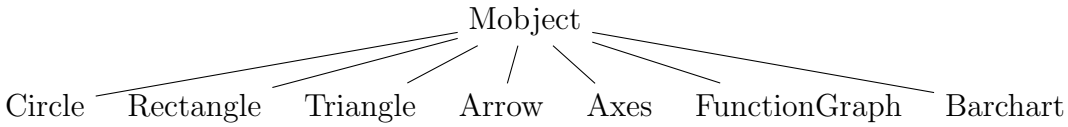
Compile and Execute C Program through CMD: Windows OS

1. Check if gcc compiler available: `gcc -v`

2. `cd Drive_name: full\path` to that folder that has the required c source file
3. Compile: `gcc c_program_name.c`
4. An executable .exe file will be created at the same directory, just run this.

Manim

Manim stands for **Mathematical Animation**. Created by **3blue1brown**. Manim is a custom python library used to animate math explanation. Manim runs on python 3.6 or higher but 3.8 is recommended. DOCS here.
Mobjects class is the root abstract class in manim. All other **Mobjects** like Circle, Rectangle, Triangles are derived from this class.



1. `manim -flags file_name.py class_name`
2. Flags
   (a) `p - play when class rendered`
   (b) `ql - low quality`
   (c) `qm - medium quality`
   (d) `qh - high quality`
   (e) `qk - 4k quality`
   (f) `a - render all scene class`
   (g) `f - open file browser location`
   (h) `i - .gif instead of .mp4`
3. Animation time by run_time: `self.play(animation(mobject), run_time = seconds)`

SQLite

SQLite is a lightweight, reliable, portable database management tool, written in C, published in 2000. Every commands(query in SQL) ends with a semicolon. We can write query in both uppercase or lowercase but writing queries in uppercase is conventional. TEXT type data allowed with both of single quote and double quote. Most like Python syntax/keywords. Not case-sensitive or case-insensitive.
Data types in SQLite:

1. NULL
2. INTEGER
3. REAL
4. TEXT
5. BLOB

SQLite has no built-in boolean types. So, we have to store boolean 0(False) & 1(True) inside of a INTEGER variable.

Queries:

1. Start query operation on a SQL file with sqlite3 engine:
   `sqlite3 file_name.sql`
   e.g. sqlite3 flights.sql

2. Get help: `.help`

3. Create table:
   `create table table_name (column_name1 data_type optional_properties, column_name2 ...);`
   e.g. create table flights (id INTEGER PRIMARY KEY AUTOINCREMENT, origin TEXT NOT NULL, destination TEXT NOT NULL, duration INTEGER NOT NULL);

4. Show all the tables exist inside the database: `.tables`

5. Insert data into table:
   `INSERT INTO table_name (comma separate columns) values(comma separated values for each corresponding column);`
   e.g. INSERT INTO flights (origin, destination, duration) values('New York', 'San Francisco', 415);

6. Get all data from a specific table in a **SQL** file: `select * from table_name;`

   (a) `select`: get/obtain/return.

   (b) `*`: all columns.

   (c) `from`: from which table.

   (d) `table_name`: specify the table in a particular SQL file.

7. Organize table in a specific mode: `.mode value`
   Values would be:

   (a) `column`: Preferred by sir Brain Yu.

   (b) `html`: Then we can copy and paste the tags and elements in a HTML document.

   (c) `box`: Boxed table, so pretty as well. My preferred.

   (d) `line`: Good one.

(e) `table`

(f) `insert`: With SQL insert statement.

(g) Or get description: `.help .mode`

8. Select all columns where rows have a particular value:
   `select * from table_name where column_name = value;`
   e.g. `select * from flights where origin = 'Tokyo';`

9. Select some columns: `SELECT column1, column2,... FROM table_name;`
   e.g. `SELECT origin, destination FROM flights;`

10. Conditional selection:
    `SELECT * FROM table_name WHERE conditions`(python's like).
    e.g. `SELECT * FROM flights WHERE duration > 300;`
    e.g. `SELECT * FROM flights WHERE duration < 200 and origin = 'New York';`
    e.g. `SELECT * FROM flights WHERE duration > 300 or destination = 'San Francisco';`
    e.g. `SELECT * FROM flights WHERE origin IN ('New York', 'San Francisco');`

11. Select by pattern matching: `SELECT * FROM flights WHERE origin like '%regex%'`
    e.g. `SELECT * FROM flights WHERE origin LIKE '%a%';`

12. Delete a row: `DELETE FROM table_name WHERE conditions;`
    e.g. `DELETE FROM flights WHERE duration > 100;`

13. Delete entire table: `DROP TABLE table_name;` e.g. `DROP TABLE flights;`

14. Exit from sqlite3 prompt console: `.quit`