Terminal Commands - A Beginner's Guide

This cheat sheet is intended to be a quick refresher for the main concepts and commands involved in using a terminal. It is not a comprehensive guide but should be enough to get the basics covered.

Contents

What is Terminal?

The terminal is a text-based interface for interacting with your computer. It allows you to perform tasks that would be difficult or impossible to do using a graphical user interface. The terminal is also known as the *command line*, *shell*, or *console*.

Why Use Terminal?

- **Efficiency**: Many tasks can be performed more quickly using a terminal than using a graphical interface.
- **Automation**: You can write scripts to automate repetitive tasks.
- **Flexibility**: A terminal gives you more control over your system than a graphical interface.
- **Remote Access**: You can connect to remote servers using a terminal.
- **Hiring**: Many technical jobs require knowledge of a terminal.
- **Cool Factor**: Using a terminal makes you look like a hacker (even if you're not).

Terminal vs Command Line vs Shell

- **Terminal**: The terminal is the program that allows you to interact with a terminal. It provides a text-based interface for running commands.
- **Command Line**: a terminal is the text-based interface itself. It allows you to type commands and receive output.
- **Shell**: The shell is the program that interprets and executes your commands. There are many different shells available, but the most common one is `bash`.

Some Keyboard Shortcuts

- **Ctrl + C**: Stop the current command and move to a clear new line.
- **Up/Down Keys**: Cycle through previous commands.
- **Tab Key**: Auto-complete file names and commands. Press twice to see all available options.
- **Ctrl + L**: Clear the terminal screen.
- **Ctrl + R**: Search through command history by typing a search term. It will show the most recent command that matches the search term.
- **Ctrl + D**: Exit the current shell.
- **q**: Quit out of a program that is running in the terminal.

Basic Commands

The following are some of the most commonly used terminal commands:

- **man [command]**

Used to show the manual/documentation for other commands. It shows all the available options, flags, and arguments for the command.

```
man ls
```

In Windows, `man` is not available. You can use `--help` instead.

```
ls --help
```

- **whoami**

  Prints the username of the current user.

  ```
  whoami # yodkwtf
  ```

- **date**

  Prints the current date and time.

  ```
  date # Fri 10 Dec 2021 10:00:00 AM IST
  ```

- **clear**

  Clears the terminal screen. Works the same as `Ctrl + L`.

  ```
  clear
  ```

File System Navigation

The following commands are used to navigate around the file system:

- **pwd**

  Prints the current working directory.

  ```
  pwd # /home/yodkwtf
  ```

  If you see `~` in the path, it represents the home directory.

- **ls [dirname]**

  Lists the files and directories within a directory. By default, it lists the files in the current directory.

  ```
  ls
  ```

  To list files in a specific directory, provide the directory name as an argument.

  ```
  ls /path/to/directory
  # ls /Downloads/
  ```

  To list all files (including hidden files), use the `-a` flag.

  ```
  ls -a
  ```

  To list files with more details (size, permissions, etc.), use the `-l` flag.

  ```
  ls -l
  ```

  To reverse the order of the list, use the `-r` flag.

  ```
  ls -r
  ```

  We can also combine flags.

  ```
  ls -ra # hidden files will be at the bottom
  ```

  There are many more flags available. You can check the manual for more information.

- **cd [dirname]**

  Changes the current directory to the specified directory.

  ```
  cd /path/to/directory
  # cd /Downloads/
  ```

  To go to the home directory, use `cd` without any arguments.

  ```
  cd
  ```

  Or use `cd ~`.

  ```
  cd ~
  ```

  To go to the last directory you were in, use `cd -`.

  ```
  cd -
  ```

  To go up one directory, use `cd ..`.

  ```
  cd ..
  ```

  To go to the root directory of the machine, use `cd /`.

  ```
  cd /
  ```

- **open [file/folder]**

  To open files, there are different commands based on the operating system.

  On MacOS, you can use `open`.

  ```
  open filename.txt
  open Downloads/
  ```

  On Linux, you can use `xdg-open`.

  ```
  xdg-open filename.txt
  xdg-open Downloads/
  ```

  On Windows, you can use `start`.

  ```
  start filename.txt
  start Downloads/
  ```

  We can also open URLs using the same command.

  ```
  open https://yodkwtf.com
  ```

File Operations

The following commands are used to perform operations on files:

- **mkdir [dirname]**

  Creates a new directory with the specified name.

  ```
  mkdir my-project
  ```

- **touch [filename]**

  Creates a new file with the specified name.

  ```
  touch filename.txt
  ```

  We can also create multiple files at once.

```
touch file1.txt file2.txt file3.txt
```

Create 100 files at once.

```
touch file-{1..100}.txt
# file-1.txt, file-2.txt, ..., file-100.txt
```

- **rm [filename]**

  Removes a file. Be careful with this command as it permanently deletes the file.

  ```
  rm filename.txt
  ```

  To get a confirmation prompt before deleting each file, use the `-i` flag.

  ```
  rm -i filename.txt
  ```

  To remove a directory, use the `-r` flag.

  ```
  rm -r directoryname
  ```

  To remove a non-empty directory without confirmation, use the `-rf` flag to force it.

  ```
  rm -rf directoryname
  ```

  > **Note**: Be very careful with the `rm -rf` command. It can delete all the files on your system if used incorrectly.

- **cp [source] [destination]**

  Copies a file from the source to the destination.

  ```
  cp file.txt /path/to/destination/
  ```

  If you want to keep the same file name, you can specify the destination directory only otherwise you can specify the new file name.

  ```
  cp file.txt /path/to/destination/
  cp file.txt /path/to/destination/newfile.txt
  ```

  To copy a directory, use the `-r` flag.

  ```
  cp -r directoryname /path/to/destination/
  ```

- **mv [source] [destination]**

  Moves a file from the source to the destination. It can also be used to rename a file.

  ```
  mv file.txt /path/to/destination/
  ```

  To rename a file, specify the new file name as the destination.

  ```
  mv file.txt newfile.txt
  ```

  To move a file and rename it, specify the new path and file name.

  ```
  mv file.txt /path/to/destination/newfile.txt
  ```

  To move a directory, use the `-r` flag.

  ```
  mv -r directoryname /path/to/destination/
  ```

- **cat [filename]**

  Displays the contents of a file.

  ```
  cat filename.txt
  ```

  To display the contents of multiple files, specify all the file names.

```
cat file1.txt file2.txt file3.txt
```

It can also be used to write to a file.

```
cat > newfile.txt # opens the file in write mode

# type the content and press Ctrl + D to save and exit
```

If the file does not exist, it will create a new file.

If the file already exists, it will overwrite the content. To append to the file, use the `>>` operator.

```
cat >> filename.txt
```

We can also use `>` (piping) to do the same things.

```
> newfile.txt
```

We can also use the `-n` flag to display line numbers.

```
cat -n filename.txt
```

These are just a few of the things `cat` can do. Check the manual for more information.

- **less [filename]**

  Displays the contents of a file one page at a time. Useful for reading large files.

  ```
  less filename.txt
  ```

  To navigate through the file, use the arrow keys or the following keys:

  - **Space**: Move forward one page.
  - **B**: Move backward one page.
  - **Q**: Quit out of the file.

  To search for a term, press `/` and type the search term.

  To exit, press `Q`.

- **head -n [lines] [filename]**

  Displays the first few lines of a file. By default, it displays the first 10 lines.

  ```
  head filename.txt
  ```

  You can specify the number of lines to display.

  ```
  head -n 20 filename.txt
  ```

- **tail -n [lines] [filename]**

  Displays the last few lines of a file. By default, it displays the last 10 lines.

  ```
  tail filename.txt
  ```

  You can specify the number of lines to display.

  ```
  tail -n 20 filename.txt
  ```

  To follow the output of a file (like logs), use the `-f` flag.

  ```
  tail -f filename.txt
  ```

  To display the last 20 lines and follow the output, use the `-n` flag.

  ```
  tail -n 20 -f filename.txt
  ```

- **nano [filename]**

Opens a file in the nano text editor.

```
nano filename.txt
```

Nano is a simple text editor that is easy to use. It shows the available commands at the bottom of the screen.

To save the file, press `Ctrl + O`.

To exit, press `Ctrl + X`.

- **echo [text]**

  Prints text to the terminal.

  ```
  echo "Hello, World!"
  ```

  To write text to a file, use the `>` operator.

  ```
  echo "Hello, World!" > filename.txt
  ```

  To append to a file, use the `>>` operator.

  ```
  echo "Hello, World!" >> filename.txt
  ```

Other Useful Commands

- **grep [pattern] [filename]**

  Searches for a pattern in a file.

  ```
  grep "pattern" filename.txt
  ```

  To search for a pattern in all files in a directory, use the `-r` flag.

  ```
  grep -r "pattern" /path/to/directory/
  ```

  To search for a pattern in a case-insensitive manner, use the `-i` flag.

  ```
  grep -i "pattern" filename.txt
  ```

  We can use Regexp patterns as well.

  ```
  grep "pattern.*" filename.txt
  ```

- **find [directory] -name [filename]**

  Finds the locations of files and directories based on conditions that you specify.

  ```
  find /path/to/directory/ -name filename.txt
  ```

  To search for a file with a specific pattern, use the `*` wildcard.

  ```
  find /path/to/directory/ -name "file-0*.txt"
  ```

  To search for a file with a specific pattern and ignore the case, use the `-iname` flag.

  ```
  find /path/to/directory/ -iname "file-0*.txt"
  ```

  To search for empty files, use the `-empty` flag.

  ```
  find /path/to/directory/ -empty
  ```

  To find and delete files, use the `-delete` flag.

  ```
  find /path/to/directory/ -name filename.txt -delete
  ```

- **piping**

Piping is a way to send the output of one command to another command.

```
ls -l | grep "file"
```

This will list all files in the current directory and then search for the word "file" in the output.

We can also use `>` to send the output to a file.

```
ls -l > files.txt
```

This will list all files in the current directory and save the output to a file called `files.txt`.

- **symlinks (symbolic links)**

  A symbolic link is a reference to another file or directory. It is similar to a shortcut in Windows.

  To create a symbolic link, use the `ln -s [srcPath] [symlinkPath]` command.

  ```
  ln -s /path/to/file /path/to/symlink
  # ln -s ~/Downloads /Desktop/dlds
  ```

  Note that it doesn't copy or move the file, it just creates a reference to it.

  On Windows, you can use `mklink` to create symbolic links.

  ```
  mklink /path/to/symlink /path/to/file
  ```

  To remove a symbolic link, use the `rm` command.

  ```
  rm /path/to/symlink
  ```

- **tar**

  The `tar` command is used to compress and decompress files.

  To create a tar archive, use the `tar -cvf [archiveName.tar] [files]` command.

  ```
  # create (-c: create, -v: verbose, -f: file)
  tar -cvf archive.tar file1.txt file2.txt
  tar -cvf archive.tar /path/to/directory/
  ```

  To see the contents of a tar archive, use the `t` flag.

  ```
  # see contents (-t: list, -v: verbose, -f: file)
  tar -tvf archive.tar
  ```

  To extract a tar archive, use the `tar -xvf [archiveName.tar]` command.

  ```
  # extract (-x: extract, -v: verbose, -f: file)
  tar -xvf archive.tar
  ```

  To extract a tar archive to a specific directory, use the `-C` flag.

  ```
  tar -xvf archive.tar -C /path/to/directory/
  ```

  To work with a compressed tar archive, use the `z` flag.

  ```
  # create (-c: create, -z: compress, -v: verbose, -f: file)
  tar -czvf archive.tar.gz file1.txt file2.txt

  # extract (-x: extract, -z: compress, -v: verbose, -f: file)
  tar -xzvf archive.tar.gz

  # see contents (-t: list, -z: compress, -v: verbose, -f: file)
  tar -tzvf archive.tar.gz
  ```

- **history**

  Displays a list of the most recent commands that you have run.

```
history
```

To search through the history, use the `grep` command.

```
history | grep "touch"
```

To run a command from the history, use `!` followed by the command number.

```
!100
```

Conclusion

If you found the cheatsheet helpful please check out more of my work on yodkwtf.com or follow me on twitter. I also run a small youtube channel called Yodkwtf Academy.