Quick *NIX shell handbook

How the shell runs commands

When a command is ran in the shell, the shells looks for the program in various places like:
/bin /usr/bin /usr/sbin /usr/local/bin /usr/games etc...

To run an executable that is in none of the places above, type a relative or absolute path to the file. Example:
```
$ ./hello
$ /home/bob/Documents/hello
```

When a command is ran it can be interrupted with CTRL-C or it can be stopped (paused) with CTRL-Z. A stopped job (process) can be resumed later with
```
$ fg    resume in the foreground
$ bg    resume in the background
```

A list of stopped jobs can be obtained with
```
$ jobs
```

Flags

A command usually takes "flags" that change the behaviour or add options. A flag starts with a hyphen: `-' followed by (normally) only one letter. Example:
```
$ ls -R
```

A flag starting with a double hyphen `--' is followed by words and is generally more meaningful to read. Example:
```
$ ls --recursive
```

Generally short flags are abbreviations of longer ones, for example the two below do the exact same thing.
```
$ ls -R
$ ls --recursive
```

Short flags can be normally (if the program is not crap) combined like so:
```
$ ls -a -h -l
$ ls -ahl
```

Basics

Typical (but not always present) help flags for any command
```
$ <CMD> --help
$ <CMD> -h
$ <CMD> -?
```

Manual for any command (if available)
```
$ man CMD
$ man -a CMD    show in succession all available manuals
```

Move around

List files and folders in a directory
```
$ ls
$ dir
$ ls -a           list all files, even one starting with `.'
$ ls -t           list and sort by (time) last change
$ ls -s -h        list files with file size
$ ls -l -h        list files and show permissions, owner, group and date
```

Change working directory
```
$ cd [PATH]
```

Print path of current working directory
```
$ pwd
```

```
Files and folders
-----------------
Move or rename a file
  $ mv FILE1 FILE2

Delete (permanently!) a file
  $ rm FILE
  $ rm -r NAME          remove a folder by recursively deleting the content
  $ rm -i NAME          ask for confirmation before deleting
  $ rm -f NAME          force delete, should NEVER be used

See space occupied by a file or folder
  $ du -h -s [NAME]

Create an empty file
  $ touch FILE

Show the content of a text file
  $ cat [FILE]
  $ tail -n N [FILE]    show only the last N lines
  $ head -n N [FILE]    show only the first N lines

Search for a pattern inside text file(s)
  $ grep PATTERN FILE [FILE2..]

Edit a file from the console
  $ nano [FILE]         simple text editor

Edit a file with a graphical editor (may not be all installed)
  $ gedit [FILE]        GNOME's default text editor
  $ leafpad [FILE]      A simple editor like Notepad on Windows
  $ kate [FILE]         KDE's default text editor
  $ subl [FILE]         Sublime Text

Create a directory (folder)
  $ mkdir DIRNAME

Delete a directory
  $ rmdir DIRNAME
  $ rm -d DIRNAME
  $ rm -rd DIRNAME      Delete a folder that is not empty by recursively
                          deleting the content of subfolders



Access permissions
------------------
In UNIX files and folder are attributed to an OWNER and a GROUP.

For the permissions 3 entities are defined: the owner, those in the group and
others. For each of the 3 entities, 3 modes are defined Read, Write and
Execute.

Note: The Execute permission makes a file executable, but when applied to a
folder allows programs like `ls' or a file manager to see the files inside that
folder.

The permissions are indicated in a string like `rwxrwxrwx'. Which means:

  User (owner)  Group  Others
  rwx           rwx    rwx

In this case everyone can do anything. When a permission is absent, the letter
is replaced with a hyphen `-', for example `rw-------' means that the file can
be edited and read only by the owner. A practical example (output of `ls -lh'):

  rw-rw----  1   john engineers  50M  Jan 03 09:12  rocket_data.dat

Is a file owned and writable by john, readable and writable by other engineers
and completely inaccessible to others.
```

```
Edit permissions
----------------
Show detailed informations about a file (i.e. access permissions)
  $ stat FILE
  $ ls -l -h FILE

Change owner of file or folder
  $ chown USER FILE
  $ chown -R USER FILE    set the owner recursively to files and subfolders

Change group of file or folder
  $ chgrp GROUP FILE
  $ chgrp -R GROUP FILE   set the group recursively to files and subfolder

Change mode of a file or folder
  $ chmod MODE FILE       see below for what MODE is
  $ chmod -R MODE FILE    set the MODE recursively to files and subfolders

  The MODE can be written in 2 ways.
    A) It can be 3 octal digits to represent the permissions by giving to each
       permission one bit. For example `rwx r-x r-x' can be `111 101 101' and
       therefore MODE is 755.

       This is used to set all permissions at once.

    B) It can specify to add or remove a permission without altering the
       existing other permissions.

       A simple notation is used where the entities are
          u = user, owner
          g = group
          o = others
          a = all, user group and others, everyone

       Followed by + or - to add or remove the permissions that are
          r = read, w = write, x = execute.

       Examples:
          - Add read and write to the owner: u+rw
          - Remove write to others: o-w
          - Allow everyone to execute: a+x

  Complete examples:
     - Set `rwxr-xr-x'
         $ chmod 755 FILE

     - Set `rwx------'
         $ chmod 700 FILE

     - Add write permissions to the group
         $ chmod g+w FILE

     - Allow everyone (owner, group, others)  to read
         $ chmod a+r FILE

     - Disallow other from writing
         $ chmod o-w FILE


File System
-----------
Perform a file system check on a disk
  # fsck /dev/sdX

Mount a disk device to /mnt
  # mount /dev/sdXn /mnt
  # mount -o ro /dev/sdXn /mnt      mount in read only
```

```
Users
-----
Print current user
  $ whoami

Print id(s) of user
  $ id             show everything UID and GIDs
  $ id [USER]
  $ id -u [USER]   show only UserID
  $ id -g [USER]   show only GroupID for your group
  $ id -G [USER]   show all GIDs to which USER belongs

Change user id or become superuser (root)
  $ su -l [USER]
  $ su

Print groups to which a user belongs
  $ groups
  $ groups [USER]

Execute as superuser (superuser do)
  $ sudo CMD

Start a shell as superuser (root) with your password (if you are administrator)
  $ sudo -s

Securely edit a file as superuser
  $ sudo -e FILE

  Warning: this command uses $EDITOR to open the file which by default is nano,
  change the editor with: $ export EDITOR=gedit


Network Stats
-------------
Show the current IP on all interfaces
  $ ifconfig         similar to Window's ipconfig
  $ ip addr          new tool to replace ifconfig

Show available network interfaces
  $ ip link

Show available network routes
  $ ip route


Internet
--------
Navigate the internet with a text only browser
  $ w3m [URL]
  $ links [URL]
  $ lynx [URL]

Download a file from an HTTP server
  $ wget URL
  $ wget -O FILENAME URL
  $ curl -o FILENAME URL

Download a file from an FTP server
  $ curl -u USERNAME URL

Remote login
------------
Connect to a remote machine with a secure shell server (sshd)
  $ ssh user@machine_ip_or_domain

Copy a file from/to a remote machine that runs a secure shell server
  $ scp user@remove:/path/of/remote/file FILE
  $ scp FILE user@remote:/path/in/remote/

Open a telnet connection
  $ telnet URL
```

```
Printing
--------
Print a file with the default printer
  $ lp FILE
  $ lpr FILE

  Here's a few useful option flags that are supported both by lp and lpr
    -o landscape
    -o media=A4
    -o sides=two-sided-long-edge
    -o sides=one-sided
    -o job-sheets=classified
    -o job-sheets=confidential
    -o page-ranges=1-4,7
    -o fit-to-page

Release a held job
   $ lp -i JOB_ID

List available printers
  $ lpstat -a

Print a file specifying the printer
  $ lp -d PRINTER FILE
  $ lpr -P PRINTER FILE

List printer jobs
  $ lpq
  $ lpstat

Remove a print job
  $ lprm JOB_ID
  $ cancel JOB_ID
```