Working with editors

Gedit

It is a graphical editor. It is simply the same as a window's notepad.
$ gedit file1.txt
$ gedit first.sh

Vi :- vi means visual editor. We can use vi to create new files and to edit content of the existing files.
$ vi file1.txt → If file1.txt is not available, then a new file will be created and opened for editing purposes.

To save this empty file we should use :wq (w means save and q means exit)

If the file already contains some data then the editor will open that file and ready for editing.

How to edit the File?

There are 3 types of modes in file editing

1) Command Mode:
   It is the default mode.
   Here we can use any vi command. From command mode, we can enter into insert mode by using multiple ways, but mostly by using it.
2) Insert Mode OR Input Mode:
   In this mode, we can modify file data. We can insert/append new data. From insert mode, we can enter into command mode by using the <Esc> key.
3) Exit Mode:
   To quit from the editor. From the command mode we have to press: then we can enter into exit mode.

Insert Mode:

How to Insert and Append Data:

A → To Append data at the end of the line
I → To Insert data at the beginning of the line
a → To append data to the right side of the cursor position (Just after the cursor position)
i → To insert data to the left side of the cursor position (Just before the cursor position
O → To open a line above the cursor position.(i.e before current line)
o → To open a line below the cursor position (i.e after current line)

How to Replace Data:

r → Replace current character.
R → To replace multiple characters from the current position.
S/cc → To replace a single line

Command Mode

How to Delete Data:

We can delete data either by:

1) character wise
    a) x → To delete a single character (del key)
    b) nx → To delete n characters.
    c) X → To delete previous character (backspace key)
    d) nX → To delete the last n previous characters.

2) word wise:
    a) dw → To delete current word.
    b) ndw → To delete n words

3) line wise:
    a) dd → To delete current line
    b) ndd → To delete n lines
    c) d$ → Deletes from current position to end of line.
    d) d^ → Deletes from current position to beginning of the line.
    e) dgg → Deletes from beginning of the file to current cursor position.
    f) Dg → Deletes from current position to end of file.

Copy and Paste Data:

yy → To Copy a Line (yanking)
Nyy → To copy N lines
yw → To copy a word
Nyw → To copy N words
y$ → To copy from current cursor position to end of line.
y^ → To copy from the beginning of the line to the current cursor position.
p → Paste above the cursor position
P → Paste below the cursor position

Cursor Navigation Commands:

k → Top Arrow
j → Bottom Arrow
l → Right Arrow
h → Left Arrow
3k → 3 Times Top Arrow

3j → 3 Times Bottom Arrow
3l → 3 Times Right Arrow
3h → 3 Times Left Arrow
$ → End of the Current Line (End Key)
^ → Beginning of the Current Line (Home Key)
H → Beginning of the Current Page
M → Middle of the Current Page
L → End of the Current Page
G → Move to last line (ie end of the file)
u → Undo last operation.
/… → Search the word after /.

Exit Mode:

:n → To go to nth line
ctrl+f → One Page Forward (Page Down)
ctrl+b → One Page Backward(Page Up)
:w → Save File Data
:wq → Save and Quit from the Editor
:q → Quit Editor
:q! → Force Quit. If we perform any changes those will be discarded.
:set nu → To set line numbers in the editor
:set nonu → To remove line numbers
:n → Place the cursor to the nth line
:$ → Place the cursor to the last line
:!<unix_command> → To execute any command.
:r!command

nano(Pico Clone) / Pico :-

It is a command line editor.
It can be used to create new files and edit content of existing files
ctrl+g (F1) Display this help text
ctrl+x (F2) Close the current file buffer / Exit from nano
ctrl+o (F3) Write the current file to disk
ctrl+r (F5) Insert another file into the current one
ctrl+w (F6) Search forward for a string or a regular expression
ctrl+\ (M-R) Replace a string or a regular expression
ctrl+k (F9) Cut the current line and store it in the cutbuffer
ctrl+u (F10) Uncut from the cut buffer into the current line
But main important options:
ctrl+o To save content
ctrl+x To quit from the editor

Vim (Vi improved)

Vim is an advanced and highly configurable text editor built to enable efficient text editing. Vim text editor is developed by Bram Moolenaar. It supports most file types and vim editor is also known as a programmer’s editor. We can use its plugin based on our needs.

file

file command is used to determine the type of a file.
Syntax: file [File name]

sed

We can use sed command to retrieve and edit data present in the given file. We can perform operations based on lines like delete 2nd line etc.
We can also perform operations based on context like delete lines where a specific word is present etc.
Syntax: Sed option expression/Script filename
Options:

1) i → in place (make actual changes in file instead of printing)
2) e → for multiple expressions. (Before all expressions have a -e option).
3) n → suppresses output. Used to print particular lines using the ‘p’ pattern.
4) r → allows regex in expression.

Note Some symbol meanings::

1) $ → represents the end of line.
2) ^ → represents the start of the file.
3) p → print specific pattern lines.
4) d → delete specific pattern lines.
5) a → Append at…
6) < > → start and end of word. To represent exact words.
7) g → for every occurrence
8) / → parts in expression
9) \ →escape character. Generally used to represent some regex or < >.
10) s → means substitute
11) + → adding some lines after matching the pattern.
12) ; → to separate multiple expression

Some Expression Examples:

1) Displaying Some Lines:
    a) sed “np” file → Display nth line two times.
    b) sed -n “3p” file → Display only 3rd line.
    c) sed -n '$p' file → Display last line.
    d) sed -n '2,4p' file → it will display from 2nd to 4th lines.
    e) sed -n ‘2!p’ file → display all except 2nd line.
    f) sed -n '2,4!p' file → it will display all lines except 2nd to 4th lines.
    g) sed -n ‘/pattern/p’ file → It will display all lines which contain the pattern.
    h) sed -n ‘2,+2p’ file → print 2nd and +2 lines.
    i) sed -n ‘ ‘/pattern/’ ,+2p’ → After matching pattern print +2 lines.
2) Delete some lines:
    a) sed ‘3d’ file → Delete the 3rd line but will not delete in the file.
    b) sed -i ‘5d’ file → Permanently deletes the 5th line.
    c) sed -i '1,$d' file → Delete all lines.
    d) sed ‘/pattern/d’ → Delete all the lines where the pattern occurs.
    e) sed ‘ /pattern/’ , +2d’ → Delete this pattern and +2 lines after that
3) Replace Some words:

a) sed ‘s/old/new/’ file → changes old word to new word for first occurrence in every line.
b) sed ‘s/old/new/i’ file → ignores case for old word.
c) sed ‘s/old/new/n’ file → changes old word to new word for nth occurrence in every line.
d) sed ‘s/old/new/g’ file → changes old word to new word for every occurrence.
e) sed ‘/r-x/ s/shubham/shub’ file → replace shubham with shub for lines where r-x substring exist.
f) sed -i 's/^$/this is a blank line/' file → replace every blank line.

4) Inserting Some lines:
a) sed ‘/pattern/a line’ → insert line after that pattern.
b) sed ‘na line’ → insert line after nth line.
c) sed ‘/pattern/i line’ → insert line before that pattern.
d) sed ‘ni line’ → insert line before nth line.