# FAP, VI Editor

## Session Objectives

- **FAP**
- **Password**
- **Floppy Access**
- **Vi editor**

# File access permissions

Since there are multiple users on a UNIX system, in order to protect individual user's files from being tampered by other users, UNIX provides a mechanism known as **file permissions**. This mechanism allows files and directories to be "owned" by a particular user.

As an example, because user1 created the files in his home directory, user1 owns those files, and has access to them.

UNIX also allows files to be shared between users and groups of users. If user1 desires, he could cut off access to his files, such that no other user could access them. However, on most systems the default is to allow other users to read your files, but not modify or delete them in any way.

As explained above, every file is owned by a particular user. However, files are also owned by a particular **group**, which is a system-defined group of users. Every user is placed into at least one group when that user is created. However, the system administrator may also grant the user access to more than one group.

Groups are usually defined by the type of users which access the machine. For example, on a university LINUX system, users may be placed into the groups student, staff, faculty or guest. There are also a few system-defined groups (such as bin and admin) which are used by the system itself to control access to resources (very rarely do actual users belong to these system groups).

Permissions fall into three main divisions:
- read
- write
- execute.

These permissions may be granted to three classes of users:
- The owner of the file
- The group to which the file belongs
- All users, regardless of group.

*Read* permission allows a user to read the contents of the file, or in the case of directories, to list the contents of the directory (using ls).

*Write* permission allows the user to write to and modify the file. For directories, write permission allows the user to create new files or delete files within that directory.

Finally, execute permission allows the user to run the file as a program or shell script (if the file happens to be a program or shell script). For directories, having *execute* permission allows the user to cd into the directory specified.

## Interpreting file permissions

Let's look at an example to demonstrate file permissions. Using the ls command with the -l option will display a ``long" listing of the file, including file permissions.

```
$ ls -l
total 20
drw-r--r--  5 user1  user1  91  May 11 15:35  abc
```

```
-rw-rw-r--  1 user1  user1  41  May 15 12:35  second
```

- The first field printed in the listing represents the file type and file permissions.
- The second field denotes the number of links.
- The third field is the owner of the file
- The fourth field is the group to which the file belongs.
- The fifth field is the size of the file in bytes.
- The sixth, seventh and eighth fields denote the day and time of last modification to the file.
- The last field is the name of the file.

The string -rw-r--r-- lists, in order, the permissions granted to the file's owner, the file's group, and everybody else.

The first character of the permissions string ("-") represents the type of file.

- A "-" just means that it is a regular file.
- A "d" just means that it is a directory file.

The next three letters ("rw-") represent the permissions granted to the file's owner, user1. The `` r'' stands for ``read'' and the `` w'' stands for ``write''. Thus, user1 has read and write permission to the file abc.

As we mentioned, besides read and write permission, there is also ``execute'' permission represented by an "x". However, there is a "-" here in place of the "x", so user1 doesn't have execute permission on this file. This is fine, the file stuff isn't a program of any kind. Of course, the owner can grant himself execute permission for the file if he so desires.

The next three characters, r--, represent the group's permissions on the file. The group, which owns this file, is users. Because only an `` r'' appears here, any user, which belongs to the group users, may read this file.

The last three characters, also r--, represent the permissions granted to every other user on the system (other than the owner of the file and those in the group users). Again, because only an `` r'' is present, other users may read the file, but not write to it or execute it.

| NOTE |
|---|
| Usually, users on a UNIX system are very open with their files. The usual set of permissions given to files is `-rw-r--r--,` which will allow other users to read the file, but not change it in any way. The usual set of permissions given to directories is `-rwxr-xr-x`, which will allow other users to look through your directories, but not create or delete files within them. |

## Changing permissions

### chmod

The `chmod` command can be used to change permissions associated with a file or directory. Only the file owner can change the file permission.

The syntax for the `chmod` command is

```
$ chmod mode file/s
```

The mode represents the permission for a type of user and can be a symbolic mode or an absolute mode.

## Symbolic mode

In the symbolic mode, the permission and the type of user for whom the permission is given are represented in symbols.

The various conventions used for symbolic mode are described in the table below:

| **Convention** | **Significance** |
|---|---|
| U | Owner of the file |
| G | Group owner |
| O | Others |
| a | All other users |
| + | Grant permission |
| - | Revoke permission |
| R | Read permission |
| W | Write permission |
| X | Execute permission |

For example, the command given below grants the execute permission to the owner of the file

```
$ chmod u+x abc
```

The following command grants the read permission to the group owner for the file abc.

```
$ chmod g+r abc
```

The command given below revokes the write permission from other users.

$  chmod o-w abc

The command given below revokes the write permission from all users.

$  chmod a-w abc

## Absolute mode

The absolute mode uses a series of digits to represent permissions. In the absolute mode it is possible to assign permissions for all types of users in one statement.

The numbers for the different permissions are

| **Permission** | **Octal representation** | **Decimal value** |
|---|---|---|
| Read (r--) | 100 | 4 |
| Write (-w-) | 010 | 2 |
| Execute (--x) | 001 | 1 |

So the command given below grants all permissions to the owner of the file, read/write permission to the group owner and read permission to others.

```
$ chmod 764 abc
```

## chown

`chown` is a standard Unix/Linux command that allows one to change the owner of a file/directory. To view the current owner of a file, execute the command `ls -l`. This will display the files in the current directory in long format. The third column in the table is the owner of the file or directory.

To change the owner of a file/directory, Type the following command:

```
$touch abc
$ chown user1 abc
```

where user1 is the new owner and abc is the name of the file or directory you want to change the owner of.

## chgrp

`chgrp` is a standard Unix/Linux command that allows one to change the group of a file/directory. To view the current group of a file, execute the following command:

`ls -l`.

This will display the files in the current directory in long format. The fourth column in the table is the group that the file/directory is associated with. To change the group of a file/directory, one must either be the owner and a member of the group one wants to change the file/directory for or one must be root.

Execute the following command:

```
$chgrp  student abc
```

where student is the new group and abc is the name of the file or directory one wants to change the owner for.

Thus, `chgrp student abc` would change the group of the file abc to student

## Password

Linux allows a measure of security by allowing the user to have a password associated with the login name. The Linux prompt appears only if both the user name and password are entered correctly. Otherwise, the user is asked to re-enter the name and password. Passwords are not displayed on the screen while they are being entered.

### Changing the password

A user can change his password with the passwd command. The following steps take place for changing the password:

```
$ passwd
Changing password for user1
(current) UNIX password:
New UNIX password:
Retype new UNIX password:
Passwd: all authentication tokens updated successfully
$_
```

In case Linux does not recognize the old password, it displays a "`passwd: Authentication failure`" message and the Linux prompt appears on the screen.

In Linux, a password should be atleast six characters long and it cannot be the same as the user's login name.

### The root user

The root user is the administrator of the Linux operating system. The administrator has all the rights necessary to control the working of the operating system. The prompt of the root user is denoted by a # sign and not the $ sign as in the case of normal users.

The root user can change the password of any user of the Linux system. He has to enter the passwd command followed by a user name and he can change the password for that user.

# VI Editor

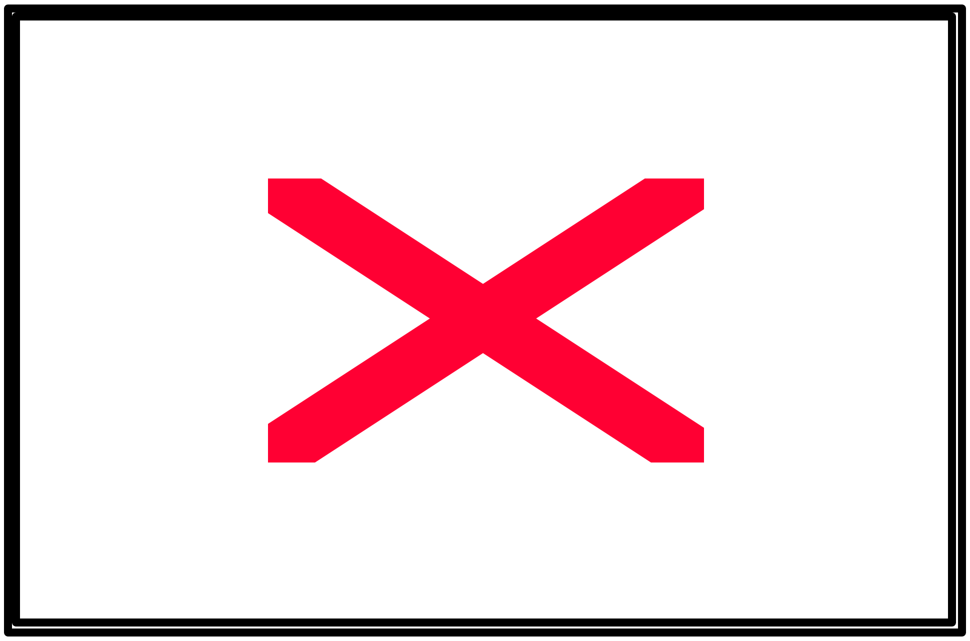
The VI editor is a screen-based editor used by many Unix users. The VI editor has powerful features to aid programmers. The VI editor lets a user create new files or edit existing files. The command to start the VI editor is `vi`, followed by the filename.

For example, to edit a file called temp, you would type

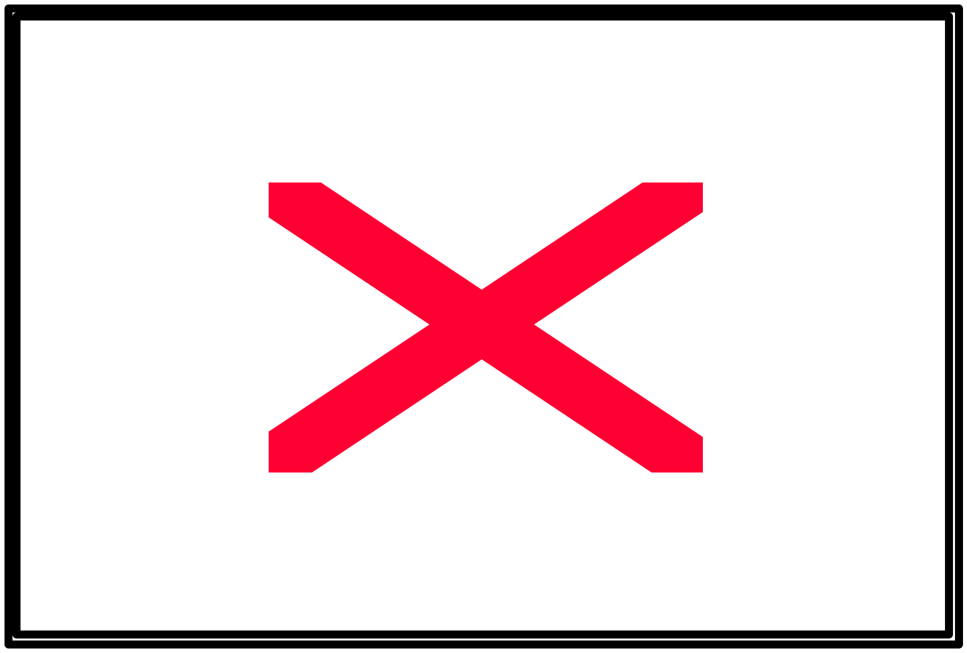
```
$ vi temporary
```

and then return. You can start VI without a filename, but when you want to save your work, you will have to tell VI which filename to save it into later.

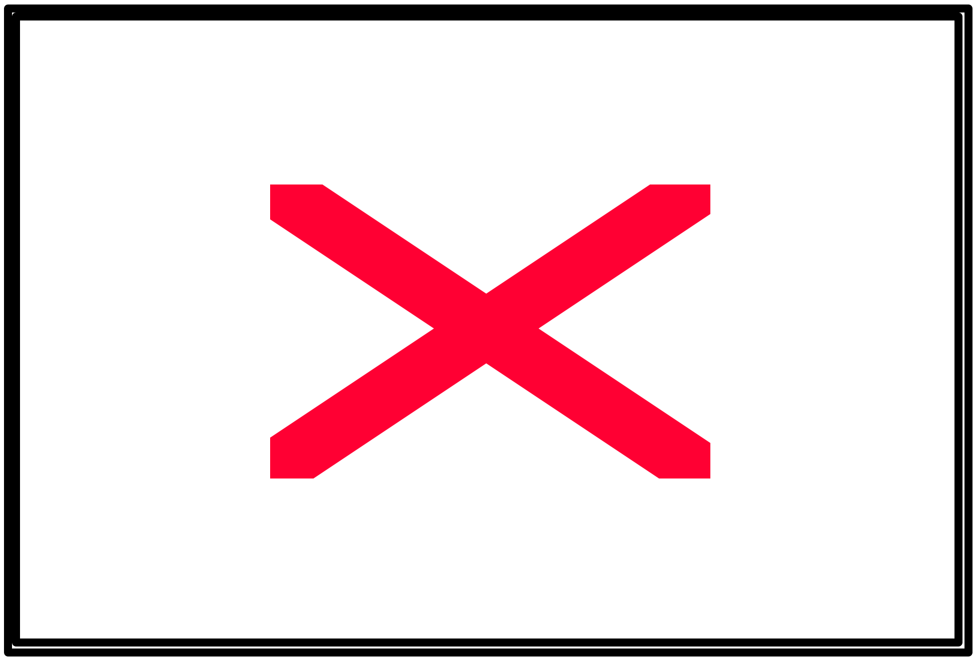
When you start VI for the first time, you will see a screen filled with tildes (~) on the left side of the screen. Any blank lines beyond the end of the file are shown this way. At the bottom of the screen, the filename should be shown, if you specified an existing file, and the size of the file will be shown as well, like this:



If the file you specified does not exist, then it will tell you that it is a new file, like this:



If you started VI without a filename, the bottom line of the screen will just be blank when VI starts.



## VI editor modes

The VI editor has three modes: *command, insert and line input mode.*

- **Command** mode allows the user to move the cursor about a text file, issue keystroke commands and change to insert mode or line input mode. Keystroke commands usually do simple things like deleting a character, word or line, moving the cursor and entering insert mode.
- **Insert** mode allows the user to type new text into the file and return to command mode.
- The **line input** mode allows the user to issue line commands. Examples include, writing edited text out to a file, searching for a text pattern or performing pattern replacements. The command line is entered by the user typing a colon "**:**". This moves the cursor to the bottom line of the screen, where the command can be typed.

VI starts out in command mode. There are several commands that put the VI editor into insert mode. The most commonly used commands to get into insert mode are **a** and **i**. These two commands are described below. Once you are in insert mode, you get out of it by hitting the escape key. You can hit Esc two times in a row and VI would definitely be in command mode. Hitting escape while you are already in command mode doesn't take the editor out of command mode. It may beep to tell you that you are already in that mode.

VI does not provide the user with the original copy of the file when invoked. It provides a editing buffer. All changes are done to that copy in the buffer. Only when the file is saved and we quit vi, the original file is replaced.

## Basic Movement

The commands used for moving about a file are:

| Keys | Action |
|---|---|
| k | Move the cursor up one line |
| | |
| j | Move the cursor down one line |
| | |
| h | Move the cursor left by one character |
| | |
| l | Move the cursor right by one character |

## Scrolling

These commands allow large vertical jumps to be made. The cursor remains still, while the text moves to a new position.

| Keys | Action |
|---|---|
| Ctrl f | scroll the screen one page forward |
| Ctrl u | scroll the screen ½ page up |
| Ctrl d | scroll the screen ½ page down |
| Ctrl b | scroll screen one page backward |

## Movement by Words

Words are recognized as blocks of alphanumeric text separated by blanks, newlines, tabs or punctuation.

| Keys | Action |
|---|---|
| w | move to start of next word (2w will move the cursor by 2 words) |
| b | move to start of previous word (6b implies move 6 words back) |
| e | move to end of next word |

These commands also have capital letter equivalents W, B & E. These have the same effect except that punctuation is not counted as a word separator.

| Keys | Action |
|---|---|
| ^ | moves to the beginning of the line |
| $ | moves to the end of the line (5$ implies move to end of fifth line) |
| L | moves to the last line of the file |
| G | moves to specific line. For example, 3G takes us to the third line |

## Adding Text

Text addition commands are issued as keystrokes from command mode. Most of them place the user in input mode.

In input mode, anything that the user types will be entered into the file. When the user hits <Esc> , vi will stop adding text and return to command mode.

The basic commands are

| Keys | Action |
|---|---|
| a | add text after the cursor |
| A | add text after end of line |
| i | insert text before cursor |
| I | insert text at start of line |
| o | open new line before cursor and insert |
| O | open new line after cursor and insert |

## Cutting

The command commonly used for cutting is d. This command deletes text from the file. The command is preceded by an optional count and followed by a movement specification. If you type dd, it deletes the current line. Here are some combinations of these:

| Keys | Action |
|---|---|
| d^ | deletes from current cursor position to the beginning of the line. |
| d$ | deletes from current cursor position to the end of the line. |
| Dw | deletes from current cursor position to the end of the word. |
| 3dd | deletes three lines from current cursor position downwards. |
| x | deletes the character at the cursor position |
| X | deletes the character before the cursor position |

## Line Input Commands

Line input mode is entered by typing a colon from command mode. The colon appears on the bottom line of the screen, and then the command can be typed there.

If the colon was typed by mistake, typing <BackSpace> will return to command mode. Otherwise, the line input command is obeyed after the user hits <Return> After the command is completed, the editor returns to command mode.

### Line numbering

Vi offers us the facility of numbering the text in a file. To do this, make sure you are in the command mode. Then press : to specify that our command is in the line input mode. Now type the following in the command line

```
:  set number
```

Line numbers are set and all lines are arranged sequentially. If we delete a line, the line nubers are changed automatically.

If we do not want the line numbering,

```
: set nonumber
```

The mode in which we are working can be identified using the command:

```
: set showmode
```

### Copy, delete and move multiple lines

To copy the third line below the sixth line, the following command can be issued:

```
: 3co6
```

To copy the first 4 lines below the eighth line, type

```
:1,4co8
```

To move the fourth line of the file to the tenth line,

```
: 4mo10
```

To move the lines 4-7 to the end of the file

```
: 4,7 mo$
```

To delete from line 5 to line 7 the following command can be used:

```
: 5,7d
```

It is also possible to delete specified lines from the current position or above the current position using notations like + and – respectively. These notations follow the **.** symbol which stands for the current cursor position.

If we want to delete the fifth line from the current cursor position, then

```
: .+4d
```

To delete the fourth line above the current cursor position:

```
: .-4d
```

## Searching

These are line-based commands. They differ from the standard ones because the user need not type a colon before typing the match pattern. Vi enters line mode to search for a pattern automatically when the `/' or `?' symbol is typed.

| | |
|---|---|
| `/abc` | search forward for the next instance of abc |
| `?abc` | search backward for the next instance of abc |
| `//` | repeat last forward search |
| `??` | repeat last backward search |

## Replacing

To replace a pattern , there are two ways of doing it.

- Replace another pattern globally

  %s/search pattern/replace pattern/g

for example to replace transys with traned, the command is

```
: %s/transys/transed/g
```

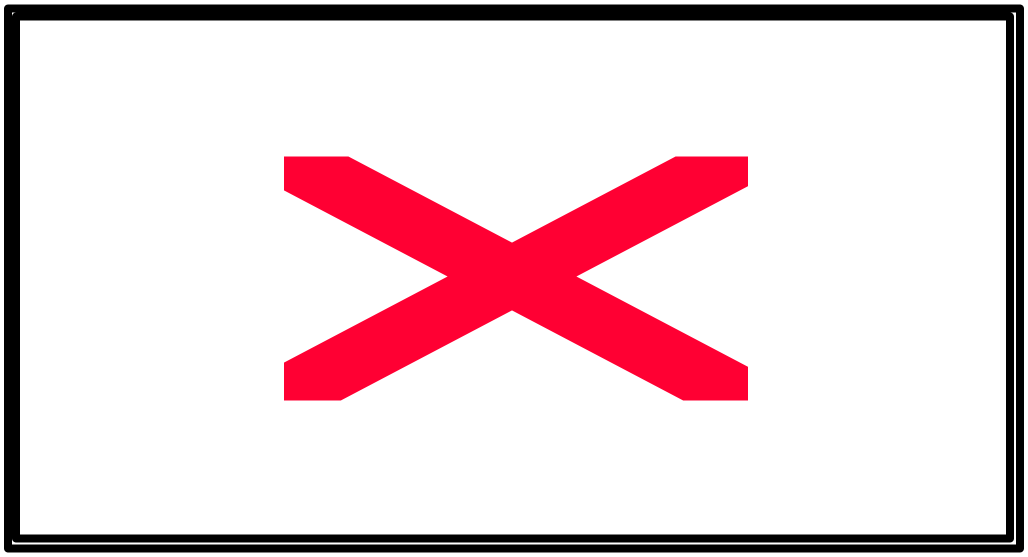
- Replace another pattern globally with confirmation

  %s/search patter/replace pattern/c
  for example to replace transys with transed, the command is

```
: %s/transys/transed/c
```

## Executing commands in VI

It is possible to execute any command in the VI editor. For example, if we want to execute the ls command inside VI, it can be done in the following manner.



**:!** is used to execute commands inside VI.

## Getting out of VI

Changes made to a file do not become permanent until you write them to the system. Commands to do this are:

| | |
|---|---|
| `:w` | write changes over the original file |
| `:w filename` | write changes to new file filename |
| `:q` | leave vi. You will not be allowed to do this unless you have saved changes. |
| `:q!` | leave the editor even if changes have not been changed |
| `:wq    or :x` | write changes and exit |
| `ZZ` | write changes and exit from the command mode |

## ☞ TOTAL RECALL

1. What are the various modes in VI?
2. How can you set line numbers for a file in VI?
3. How can a user come out of VI?

## REVIEW QUESTIONS

1. A __________ sign denotes the prompt of the root user.
2. A user can change his password with the _____________ command.
3. The __________ and __________ symbols can be used to indicate granting and revoking rights for a file respectively.
4. The permissions for a particular file can be changed using the ____________ command.
5. The two modes of granting and revoking permissions are ____________________.
6. To copy the third line below the fifth line, the command is __________________.
7. The ______________ key can be used to delete a word of text.
8. To quit after saving the current changes to the file, we can use _______________.