Arithmetic Operators in VI Editor:

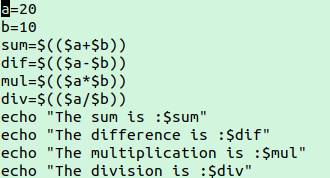
The VI editor is known for its versatility across the tech world. We can perform numerous operations in it. Here we are going to explore the arithmetic operations that can be performed in the VI editor. For learning them, first we should be familiar with all the commands we are going to use in it.

Commands to be used:

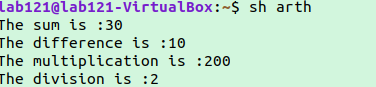
* Echo: echo command in linux is used to display line of text/string that are passed as an argument.
* Read: The command read in the Linux is used to read the input from the keyboard.
* Sh : It is a command language interpreter that executes commands read from a [command line](https://www.computerhope.com/jargon/c/commandi.htm) [string](https://www.computerhope.com/jargon/s/string.htm), the [standard input](https://www.computerhope.com/jargon/s/stdin.htm), or a specified [file](https://www.computerhope.com/jargon/f/file.htm).

Addition, subtraction, multiplication and division is performed as stated.

The following screenshot is of the vi editor file named ‘arth’ I created:



The output is as follows:



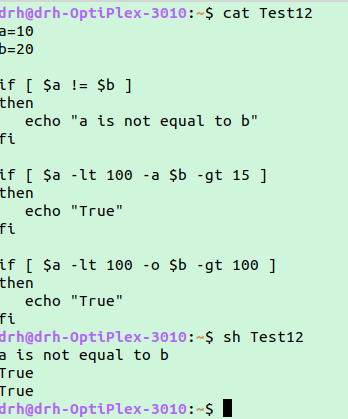
Logical Operations in VI editor:

For this first we must know the logical operators which are going to be used.

* ! : This is logical negation. This inverts a true condition into false and vice versa.
* -o : This is logical OR. If one of the operands is true, then the condition becomes true.
* -a : This is logical AND. If both the operands are true, then the condition becomes true otherwise false.

In the program I have used 2 more functionalities, -gt stands for greater than and -lt stands for less then.

Here is the screenshot:



File Permissions in Linux:

Being a multi-user operating system the crucial aim of Linux should be to control the access of information by different users. For effective security, Linux divides authorization into 2 levels.

1. Ownership
2. Permission

* Ownership of Linux files:

Every file and directory on your Unix/Linux system is assigned 3 types of owner, given below.

User

A user is the owner of the file. By default, the person who created a file becomes its owner. Hence, a user is also sometimes called an owner.

Group

A user- group can contain multiple users. All users belonging to a group will have the same access permissions to the file. Suppose you have a project where a number of people require access to a file. Instead of manually assigning permissions to each user, you could add all users to a group, and assign group permission to file such that only this group members and no one else can read or modify the files.

Other

Any other user who has access to a file. This person has neither created the file, nor he belongs to a usergroup who could own the file. Practically, it means everybody else. Hence, when you set the permission for others, it is also referred as set permissions for the world.

## Permissions

**We can have following permissions:**

**Read:** This permission gives you the authority to open and read a file. Read permission on a directory gives you the ability to lists its content.

**Write:** The write permission gives you the authority to modify the contents of a file. The write permission on a directory gives you the authority to add, remove and rename files stored in the directory.

**Execute:** In Windows, an executable program usually has an extension ".exe" and which you can easily run. In Unix/Linux, you cannot run a program unless the execute permission is set. If the execute permission is not set, you might still be able to see/modify the program code (provided read & write permissions are set), but not run it.

## Changing file/directory permissions with 'chmod' command

We can use the '**chmod'** command which stands for 'change mode'. Using the command, we can set permissions (read, write, execute) on a file/directory for the owner, group and the world. **Syntax:**

chmod permissions filename

There are 2 ways to use the command -

1. **Absolute mode**
2. **Symbolic mode**

## Absolute(Numeric) Mode

In this mode, file **permissions are not represented as characters but a three-digit octal number**.

The table below gives numbers for all for permissions types.

|  |  |  |
| --- | --- | --- |
| **Number** | **Permission Type** | **Symbol** |
| 0 | No Permission | --- |
| 1 | Execute | --x |
| 2 | Write | -w- |
| 3 | Execute + Write | -wx |
| 4 | Read | r-- |
| 5 | Read + Execute | r-x |
| 6 | Read +Write | rw- |
| 7 | Read + Write +Execute | rwx |

## Symbolic Mode

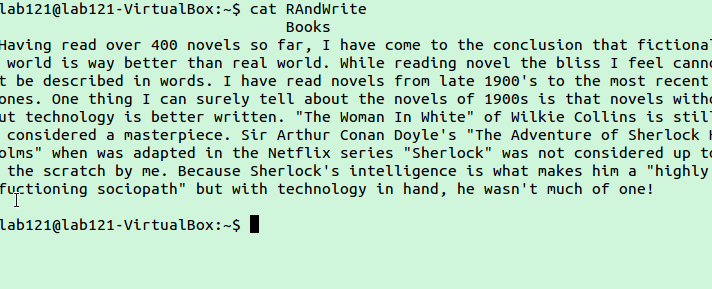
In the Absolute mode, you change permissions for all 3 owners. In the symbolic mode, you can modify permissions of a specific owner. It makes use of mathematical symbols to modify the file permissions.

|  |  |
| --- | --- |
| **Operator** | **Description** |
| **+** | Adds a permission to a file or directory |
| **-** | Removes the permission |
| **=** | Sets the permission and overrides the permissions set earlier. |

The various owners are represented as -

|  |  |
| --- | --- |
| **User Denotations** | |
| u | user/owner |
| g | group |
| o | other |
| a | all |

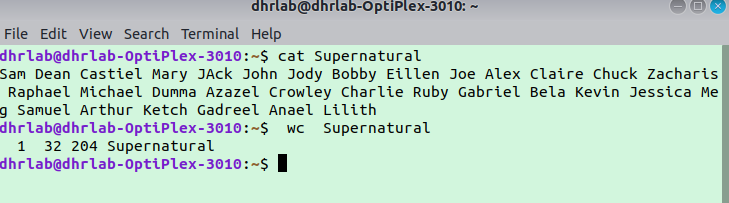
Let’s see both the ways in action:





Word Count of a file in VI Editor:

The **wc** (**word count**) command in Unix/Linux operating systems is used to find out number of **newline count**, **word count**, **byte and characters** count in a files specified by the file arguments.



Finding greatest and smallest number from a given set:

In this we make smart use of operators to find the smallest and greatest number from the input numbers given by the user.

