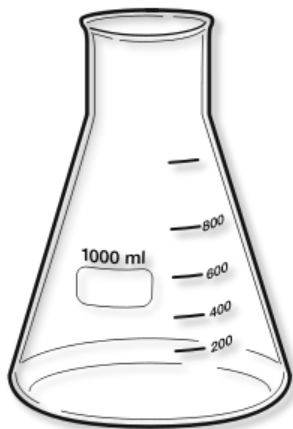


SecureSet Academy



{LAB} Intro to Linux

Created by ajay Menendez [@paladin63](#)

PREREQUISITES:

- **Host Based Virtualization - Virtualbox**

Basic Understanding on how to use Host based Virtualization, specifically Virtual Box.

If you are unfamiliar please participate in SecureSet's Virtualization Hacking 101 in advance of conducting this lab.

OBJECTIVE:

Today's lab objective is to be able to begin with the basics of Linux and working in its command line interface.

BEFORE YOU START:

1. Command Line Basics

- a. The bash shell is one of several shells available for Linux.
- b. It is also called the Bourne-again shell, after Stephen Bourne, the creator of an earlier shell (/bin/sh).
- c. Bash is substantially compatible with sh, but it provides many improvements in both function and programming capability.
- d. No this has nothing to do with Jason Bourne.

2. To OPEN a TERMINAL (If you are running a Desktop version of Linux) (Otherwise you will start in the terminal)

- a. If you're running Unity: open the dash, type terminal, hit Return.
- b. If you're on the old-style menus, Applications → Accessories → Terminal.
- c. **Control + Alt + T.**

3. What is a command?

- a. **A simple command** consists of a sequence of words separated by spaces or tabs. The first word is taken to be the name of a command, and the remaining words are passed as arguments to the command.
- b. Before we delve deeper into bash, recall that a shell is a program that accepts and executes commands.
- c. It also supports programming constructs, allowing complex commands to be built from smaller parts.
- d. These complex commands, or scripts, can be saved as files to become new commands in their own right.
- e. Many commands on a typical Linux system are scripts.

4. **Shells** have some built-in commands, such as cd, break, and exec.

- a. Other commands are external. We will review most of the commands shortly, however we need to address I/O first.

5. **Shells** also use three standard I/O streams:

- a. the standard input stream, which provides input to commands is called **stdin**
- b. the standard output stream, which displays output from commands is called **stdout**

- c. the standard error stream, which displays error output from commands is called **stderr**
 - d. Input streams provide input to programs, usually from terminal keystrokes. Output streams print text characters, usually to the terminal. The terminal was originally an ASCII typewriter or display terminal, but it is now more often a window on a graphical desktop.
6. In order to understand about **Redirecting** operators in Linux we should know how we communicate with a computer.
7. When we are communicating with a computer there should be a way to do it and this is achieved by STDIN (0), STDOUT (1), STDERR (2) file descriptors.
- a. These **file descriptors** are assigned with a number as shown below:
 - 1. STDIN → **0**
 - 2. STDOUT → **1**
 - 3. STDERR → **2**
8. **STDIN**: stands for **STandarD INput**.
- a. By using this we can give an input to the computer to do some task. Whatever device we used to give input to a computer will come under **STDIN**.
 - b. A **STDIN** can be A keyboard A mouse A scanner A floppy A CD/DVD ROM A touch screen A barcode reader or a card reader.
9. **STDOUT**: The abbreviation is **STandarD OUTput**.
- a. By using this we can see the output from a computer for the it just processed or executed. Whatever device we used to get the output from a computer will come under this **STDOUT**.
 - b. A **STDOUT** can be A monitor A speaker Or a printer
10. **STDERR**: This is abbreviated as **STandarD ERRor**.
- a. By using this, A computer can communicate with user to give him warning/error etc. that something went wrong when executing a task.

- b. A **STDERR** can be A monitor A printer A log file A LED indicator Or a speaker.

11. Shells 2.0

- a. Every user has a unique username. When they logon to the system, they are placed in a HOME directory, which is a portion of the disk space reserved just for them.
- b. When you log onto a UNIX system, your main interface to the system is called the UNIX SHELL.
- c. This is the program that presents you with the dollar sign (\$) prompt.
- d. This prompt means that the shell is ready to accept your typed commands.
- e. We know in Ubuntu Server or Desktop you will using one of the most standard UNIX shells called the **Bourne Shell**.
- f. UNIX commands are strings of characters typed in at the keyboard. To run a command, you just type it in at the keyboard and press the ENTER key.
- g. **UNIX extends the power of commands by using special flags, arguments or switches.**
- h. **These operators are one of the most powerful features of UNIX commands.**
- i. Switches are usually preceded with a dash (-) and precede any filenames or other arguments on the command line.
- j. Unlike the DOS (or Windows) command line, UNIX systems are case sensitive (upper and lower-case characters are considered different).
- k. Nearly all command names and most of their command line switches will be in lowercase.
- l. In this class all of the UNIX commands should be typed in lowercase characters unless explicitly instructed otherwise.
- m. UNIX systems have a hierarchical directory structure. This means that the hard disk area is divided into directories, much like a book is sub-divided into chapters and paragraphs. The directories form a tree-like structure, which simplifies the organization of the files on the system.

12. An abstraction is the process of taking away or removing characteristics from something in order to reduce it to a set of essential characteristics.

- a. In Computer file systems, a file is a singular object.
- b. A folder is a container for those objects.
 - 1. You can have tons, millions of objects in a **container.** Which is a folder.
- c. You can have folders *IN* folders creating a hierarchy. They can be folders in folders in folders. Get the idea?
- d. It's rather like a group of Russian Babushka Nesting dolls.

13. It's important to understand this relationship because we are about to embark on the tale of the **FILE SYSTEM.**

14. **LINUX FILE SYSTEM -- This is a simplified view**

- a. Linux Saying
 - 1. **"On a UNIX system, everything is a file; if something is not a file, it is a process."**
 - 2. This statement is true because there are special files that are more than just files but to keep things simple, saying that everything is a file is an acceptable generalization. A Linux system, just like UNIX, makes no difference between a file and a directory, since a directory is just a file containing names of other files. Programs, services, texts, images, and so forth, are all files. Input and output devices, and generally all devices, are considered to be files, according to the system.
 - 3. In order to manage all those files in an orderly fashion, man likes to think of them in an ordered tree-like structure on the hard disk, as we know from MS-DOS (Disk Operating System) for instance. The large branches contain more branches, and the branches at the end contain the tree's leaves or normal files.
- b. For now, we will use an image of the tree, to wrap our heads around this.

- c. Depending on the system admin, the operating system and the mission of the UNIX machine, the structure may vary, and directories may be left out or added at will. The names are not even required; they are only a convention.

15. CLI START

- a. When you first log into a UNIX system, you are placed in your own personal directory space called your HOME directory.
 - 1. On most UNIX systems, the user HOME directories are located under the /home directory.
- b. Types of prompts
 - 1. user@computername~\$
 - 2. ajay@linux1:~\$ Home Directory
 - 3. ajay@linux1:/\$ Root of File System
 - 4. ajay@linux1:/home Home folder, but not my user home folder. <Know the structure! This is a trick>

LINUX HACKING 101 LAB

PLEASE NOTE!

ALL TYPED OUT COMMANDS to type are in RED, things will the need for special attention will be in **Yellow Highlighter**.

If you have never used the Linux Command Line, it would be best if you reviewed these helpful webpages at one point:

<https://developer.ibm.com/tutorials/l-lpic1-103-1/>

<https://www.gnu.org/software/bash/manual/bashref.html#Redirections>

<http://www.linfo.org/argument.html>

<https://www.w3resource.com/linux-system-administration/commands-and-arguments.php>

PreRequisites:

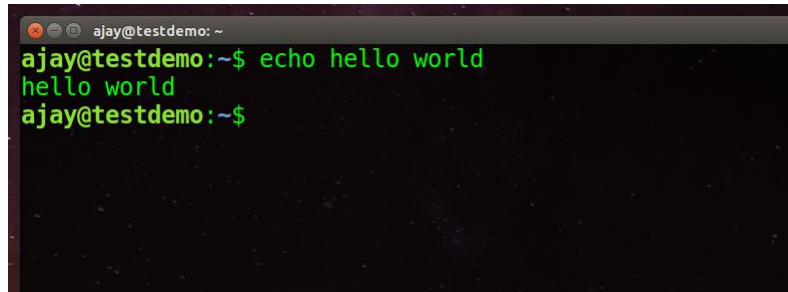
- Please have a version of Ubuntu Desktop Linux Installed.
 - Or have Ubuntu Desktop Linux ready by whatever means.
- Please have a version of Oracle Virtual Box Installed.

~If~ you don't, you will need to install Virtual Box and then you will want to download this VDI (Virtual Disk) from the following LINK: Download the VDI file from <URL>

LAB START:

echo hello world

Expected output:

A terminal window with a dark background and green text. The prompt is 'ajay@testdemo: ~'. The command 'echo hello world' is entered, and the output 'hello world' is displayed on the next line. The prompt 'ajay@testdemo:~\$' is shown again.

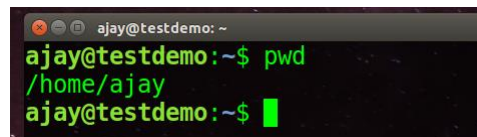
```
ajay@testdemo:~$ echo hello world
hello world
ajay@testdemo:~$
```

[https://en.wikipedia.org/wiki/%22Hello, World!%22_program](https://en.wikipedia.org/wiki/%22Hello,_World!%22_program)

A "Hello, World!" program generally is a computer program that outputs or displays the message "Hello, World!". Because it is very simple in most programming languages, it is often used to illustrate the basic syntax of a programming language and is often the first program that those learning to code write.

pwd

Expected output:

A terminal window with a dark background and green text. The prompt is 'ajay@testdemo: ~'. The command 'pwd' is entered, and the output '/home/ajay' is displayed on the next line. The prompt 'ajay@testdemo:~\$' is shown again.

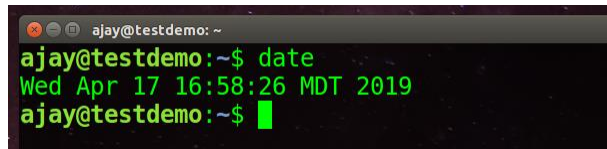
```
ajay@testdemo:~$ pwd
/home/ajay
ajay@testdemo:~$
```

<https://www.howtoforge.com/linux-pwd-command/>

The pwd command, like ls and cd, is one of the most frequently used Linux utilities. Regardless of the kind of user you are (newbie or pro), you'll find yourself using this command line tool a lot. The pwd tool prints the name of the present/current working directory (PWD - Present Working Directory, got it?).

date

Expected output: (Please note your time will be current, not what I have in this output)

A terminal window titled 'ajay@testdemo: ~' showing the command 'date' being executed. The output is 'Wed Apr 17 16:58:26 MDT 2019'. The prompt 'ajay@testdemo:~\$' is visible on the line above and below the output.

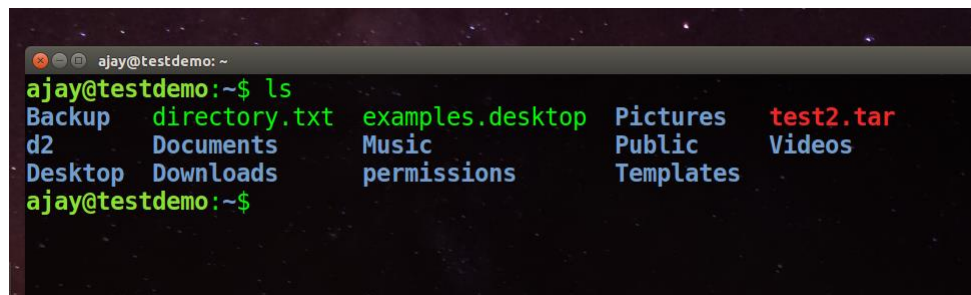
```
ajay@testdemo:~$ date
Wed Apr 17 16:58:26 MDT 2019
ajay@testdemo:~$
```

<https://www.linode.com/docs/tools-reference/tools/use-the-date-command-in-linux/>

The date command displays the current date and time. It can also be used to display or calculate a date in a format you specify.

ls

Expected output: (Please note your directory might be slightly different)

A terminal window titled 'ajay@testdemo: ~' showing the command 'ls' being executed. The output lists files and directories in a color-coded format: Backup, d2, Desktop, directory.txt, Documents, Downloads, examples.desktop, Music, permissions, Pictures, Public, Templates, test2.tar, and Videos.

```
ajay@testdemo:~$ ls
Backup  directory.txt  examples.desktop  Pictures  test2.tar
d2      Documents      Music             Public    Videos
Desktop Downloads      permissions       Templates
ajay@testdemo:~$
```

<https://linuxize.com/post/how-to-list-files-in-linux-using-the-ls-command/>

More on ls: https://www.ibm.com/support/knowledgecenter/en/SSLTBW_2.3.0/com.ibm.zos.v2r3.bpxa500/lscmd.htm

The ls command is one of the basic commands that any Linux user should know. It is used to list information about files and directories within the file system. The ls utility is a part of the GNU core utilities which are installed on all Linux distributions.

How to Use the ls Command

The syntax for the ls command is as follows:

ls [OPTIONS] [FILES]

When used with no arguments, the ls command will list the names of all files in the current working directory: **ls**

The files are listed in alphabetical order: cache db empty games lib local lock log mail opt run pool tmp

To list files in a specific directory, pass the path to the directory to the ls command. For example, to list the contents of the /etc directory, type:

ls /etc

ls -l

Expected output: (Please note your directory might be slightly different)

```
ajay@testdemo: ~$ ls -l
total 92
drwxrwxr-x 2 ajay ajay 4096 Nov  4 10:58 Backup
drwxrwxr-x 2 ajay ajay 4096 Nov  5 11:18 d2
drwxr-xr-x 2 ajay ajay 4096 Oct 29 12:59 Desktop
-rw-rw-r-- 1 ajay ajay  244 Apr 15 15:18 directory.txt
drwxr-xr-x 3 ajay ajay 4096 Mar 27 12:43 Documents
drwxr-xr-x 2 ajay ajay 4096 Mar 27 14:55 Downloads
-rw-r--r-- 1 ajay ajay 8980 Oct 29 12:56 examples.desktop
drwxr-xr-x 2 ajay ajay 4096 Oct 29 12:59 Music
drwxrwxr-x 2 ajay ajay 4096 Nov  1 12:58 permissions
drwxr-xr-x 2 ajay ajay 4096 Oct 29 12:59 Pictures
drwxr-xr-x 2 ajay ajay 4096 Oct 29 12:59 Public
drwxr-xr-x 2 ajay ajay 4096 Oct 29 12:59 Templates
-rw-rw-r-- 1 ajay ajay 30720 Nov  5 11:18 test2.tar
drwxr-xr-x 2 ajay ajay 4096 Oct 29 12:59 Videos
ajay@testdemo: ~$
```

Long Listing Format

The default output of the ls command shows only the names of the files, which is not very informative. When the long listing format is used the ls command will display the following file information:

- The file type
- The file permissions
- Number of hard links to the file
- File owner
- File group
- File size
- Date and Time
- File name

The -l (lowercase L) option causes ls to print files in long listing format.

ls -l /etc/hosts

```
-rw-r--r-- 1 root root 337 Oct  4 11:31 /etc/hosts
```

Notice that the highlighted red box shows, that it is giving you the size of files and folders in BYTES.

(More on COMPUTER FILE SIZES: <http://myrepono.com/faq/4>)

ls -lh

Expected output: (Please note your directory might be slightly different)

```
ajay@testdemo:~$ ls -lh
total 92K
drwxrwxr-x 2 ajay ajay 4.0K Nov  4 10:58 Backup
drwxrwxr-x 2 ajay ajay 4.0K Nov  5 11:18 d2
drwxr-xr-x 2 ajay ajay 4.0K Oct 29 12:59 Desktop
-rw-rw-r-- 1 ajay ajay 244 Apr 15 15:18 directory.txt
drwxr-xr-x 3 ajay ajay 4.0K Mar 27 12:43 Documents
drwxr-xr-x 2 ajay ajay 4.0K Mar 27 14:55 Downloads
-rw-r--r-- 1 ajay ajay 8.8K Oct 29 12:56 examples.desktop
drwxr-xr-x 2 ajay ajay 4.0K Oct 29 12:59 Music
drwxrwxr-x 2 ajay ajay 4.0K Nov  1 12:58 permissions
drwxr-xr-x 2 ajay ajay 4.0K Oct 29 12:59 Pictures
drwxr-xr-x 2 ajay ajay 4.0K Oct 29 12:59 Public
drwxr-xr-x 2 ajay ajay 4.0K Oct 29 12:59 Templates
-rw-rw-r-- 1 ajay ajay 30K Nov  5 11:18 test2.tar
drwxr-xr-x 2 ajay ajay 4.0K Oct 29 12:59 Videos
ajay@testdemo:~$
```

You will notice in this example, I'm using ls -l but adding a h, so it will be ls-lh

ls = LIST

-l = long

-h = human readable

ls -lh = List -> Long format + Human Readable

You will notice that instead of giving it to you in BYTES it is giving it to you in KILOBYTES, MEGABYTES, or GIGABYTES as needed.

ls -la

Expected output: (Please note your directory might be slightly different)

```
ajay@testdemo:~$ ls -la
total 172
drwxr-xr-x 19 ajay ajay 4096 Apr 17 16:49 .
drwxr-xr-x  4 root root 4096 Nov  1 12:08 ..
drwxrwxr-x  2 ajay ajay 4096 Nov  4 10:58 Backup
-rw-----  1 ajay ajay 5901 Apr 15 15:26 .bash_history
-rw-r--r--  1 ajay ajay  220 Oct 29 12:56 .bash_logout
-rw-r--r--  1 ajay ajay 3771 Oct 29 12:56 .bashrc
drwx----- 17 ajay ajay 4096 Apr 15 14:50 .cache
drwx----- 17 ajay ajay 4096 Oct 29 13:17 .config
drwxrwxr-x  2 ajay ajay 4096 Nov  5 11:18 d2
drwxr-xr-x  2 ajay ajay 4096 Oct 29 12:59 Desktop
-rw-rw-r--  1 ajay ajay  244 Apr 15 15:18 directory.txt
-rw-r--r--  1 ajay ajay   25 Oct 29 12:59 .dmrc
drwxr-xr-x  3 ajay ajay 4096 Mar 27 12:43 Documents
drwxr-xr-x  2 ajay ajay 4096 Mar 27 14:55 Downloads
-rw-r--r--  1 ajay ajay 8980 Oct 29 12:56 examples.desktop
drwx-----  2 ajay ajay 4096 Oct 29 13:07 .gconf
drwx-----  3 ajay ajay 4096 Apr 17 16:49 .gnupg
-rw-----  1 ajay ajay 3594 Apr 17 16:49 .ICEauthority
drwx-----  3 ajay ajay 4096 Oct 29 12:59 .local
drwx-----  5 ajay ajay 4096 Oct 29 13:04 .mozilla
drwxr-xr-x  2 ajay ajay 4096 Oct 29 12:59 Music
drwxrwxr-x  2 ajay ajay 4096 Nov  1 12:58 permissions
drwxr-xr-x  2 ajay ajay 4096 Oct 29 12:59 Pictures
-rw-r--r--  1 ajay ajay  655 Oct 29 12:56 .profile
drwxr-xr-x  2 ajay ajay 4096 Oct 29 12:59 Public
-rw-r--r--  1 ajay ajay    0 Oct 29 13:17 .sudo_as_admin_successful
drwxr-xr-x  2 ajay ajay 4096 Oct 29 12:59 Templates
-rw-rw-r--  1 ajay ajay 30720 Nov  5 11:18 test2.tar
drwxr-xr-x  2 ajay ajay 4096 Oct 29 12:59 Videos
-rw-----  1 ajay ajay 5422 Nov  5 13:40 .viminfo
-rw-----  1 ajay ajay   53 Apr 17 16:49 .Xauthority
-rw-----  1 ajay ajay   82 Apr 17 16:49 .xsession-errors
-rw-----  1 ajay ajay 1363 Apr 15 15:55 .xsession-errors.old
ajay@testdemo:~$
```

Show Hidden Files

By default, the ls command will not show hidden files.

In Linux, a hidden file is any file that begins with a dot (.).

To display all files including the hidden files use the -a option:

ls -la

Please note you can mix and match the arguments for the ls command.

PROTIP: If you attend any of SecureSet's CTF (Capture the Flags, this might come in handy.)

mkdir

Expected output:

```
ajay@testdemo: ~  
ajay@testdemo:~$ mkdir test  
ajay@testdemo:~$  
ajay@testdemo:~$ ls -l  
total 96  
drwxrwxr-x 2 ajay ajay 4096 Nov  4 10:58 Backup  
drwxrwxr-x 2 ajay ajay 4096 Nov  5 11:18 d2  
drwxr-xr-x 2 ajay ajay 4096 Oct 29 12:59 Desktop  
-rw-rw-r-- 1 ajay ajay  244 Apr 15 15:18 directory.txt  
drwxr-xr-x 3 ajay ajay 4096 Mar 27 12:43 Documents  
drwxr-xr-x 2 ajay ajay 4096 Mar 27 14:55 Downloads  
-rw-r--r-- 1 ajay ajay 8980 Oct 29 12:56 examples.desktop  
drwxr-xr-x 2 ajay ajay 4096 Oct 29 12:59 Music  
drwxrwxr-x 2 ajay ajay 4096 Nov  1 12:58 permissions  
drwxr-xr-x 2 ajay ajay 4096 Oct 29 12:59 Pictures  
drwxr-xr-x 2 ajay ajay 4096 Oct 29 12:59 Public  
drwxr-xr-x 2 ajay ajay 4096 Oct 29 12:59 Templates  
drwxrwxr-x 2 ajay ajay 4096 Apr 17 17:09 test  
-rw-rw-r-- 1 ajay ajay 30720 Nov  5 11:18 test2.tar  
drwxr-xr-x 2 ajay ajay 4096 Oct 29 12:59 Videos  
ajay@testdemo:~$
```

<http://www.linfo.org/mkdir.html>

The mkdir command is used to create new directories. A directory, referred to as a folder in some operating systems, appears to the user as a container for other directories and files.

rmdir

Expected output:

```
ajay@testdemo: ~  
ajay@testdemo:~$ rmdir test  
ajay@testdemo:~$ ls -l  
total 92  
drwxrwxr-x 2 ajay ajay 4096 Nov  4 10:58 Backup  
drwxrwxr-x 2 ajay ajay 4096 Nov  5 11:18 d2  
drwxr-xr-x 2 ajay ajay 4096 Oct 29 12:59 Desktop  
-rw-rw-r-- 1 ajay ajay  244 Apr 15 15:18 directory.txt  
drwxr-xr-x 3 ajay ajay 4096 Mar 27 12:43 Documents  
drwxr-xr-x 2 ajay ajay 4096 Mar 27 14:55 Downloads  
-rw-r--r-- 1 ajay ajay 8980 Oct 29 12:56 examples.desktop  
drwxr-xr-x 2 ajay ajay 4096 Oct 29 12:59 Music  
drwxrwxr-x 2 ajay ajay 4096 Nov  1 12:58 permissions  
drwxr-xr-x 2 ajay ajay 4096 Oct 29 12:59 Pictures  
drwxr-xr-x 2 ajay ajay 4096 Oct 29 12:59 Public  
drwxr-xr-x 2 ajay ajay 4096 Oct 29 12:59 Templates  
-rw-rw-r-- 1 ajay ajay 30720 Nov  5 11:18 test2.tar  
drwxr-xr-x 2 ajay ajay 4096 Oct 29 12:59 Videos  
ajay@testdemo:~$
```

<https://www.computerhope.com/unix/urmdir.htm>

The rmdir command removes each directory specified on the command line, **if they are empty**.

That is, each directory removed must contain no files or directories, or it cannot be removed by rmdir.

If any specified directory is not empty, rmdir will not remove it, and will proceed to try and remove any other directories you specified.

LET'S TAKE IT TO THE NEXT LEVEL

COMMAND we are learning and working with is “cd”

mkdir test

cd test

pwd

Expected output: (Please note your directory and path might be slightly different)

A terminal window with a dark background and green text. The prompt is 'ajay@testdemo: ~/test'. The commands and their outputs are: 'mkdir test' (no output), 'cd test' (no output), and 'pwd' (output: '/home/ajay/test').

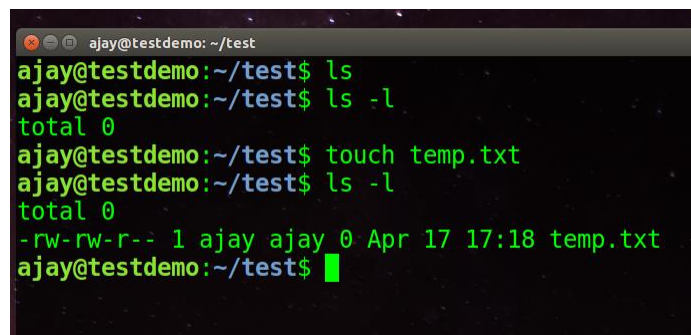
```
ajay@testdemo: ~/test
ajay@testdemo:~$ mkdir test
ajay@testdemo:~$ cd test
ajay@testdemo:~/test$ pwd
/home/ajay/test
ajay@testdemo:~/test$
```

<https://www.computerhope.com/jargon/c/cd.htm>

Short for Change Directory, cd is a command commonly used to change the directory in a command line operating system.

touch

Expected output:

A terminal window with a dark background and green text. The prompt is 'ajay@testdemo: ~/test'. The commands and their outputs are: 'ls' (output: 'total 0'), 'ls -l' (output: 'total 0'), 'touch temp.txt' (no output), and 'ls -l' (output: '-rw-rw-r-- 1 ajay ajay 0 Apr 17 17:18 temp.txt').

```
ajay@testdemo: ~/test
ajay@testdemo:~/test$ ls
total 0
ajay@testdemo:~/test$ ls -l
total 0
ajay@testdemo:~/test$ touch temp.txt
ajay@testdemo:~/test$ ls -l
total 0
-rw-rw-r-- 1 ajay ajay 0 Apr 17 17:18 temp.txt
ajay@testdemo:~/test$
```

<https://www.computerhope.com/unix/utouch.htm>

touch changes file timestamps. It is also an easy way to create empty files. (Which is what we were using it for)

`cp temp.txt temp2.txt`

Expected output:

```
ajay@testdemo: ~/test
ajay@testdemo:~/test$ ls
temp.txt
ajay@testdemo:~/test$ cp temp.txt temp2.txt
ajay@testdemo:~/test$ ls
temp2.txt temp.txt
ajay@testdemo:~/test$
```

<https://www.cyberciti.biz/faq/copy-command/>

To copy files and directories use the `cp` command under a Linux, UNIX-like, and BSD like operating systems. `cp` is the command entered in a Unix and Linux shell to copy a file from one place to another, possibly on a different filesystem. The original file remains unchanged, and the new file may have the same or a different name.

In the above example, we are duplicating the same file w/ a different name in the same directory.

`cp temp2.txt /home/ajay/Documents/`

Expected output:

```
ajay@testdemo: ~/Documents
ajay@testdemo:~/test$ ls
temp.txt
ajay@testdemo:~/test$ cp temp.txt temp2.txt
ajay@testdemo:~/test$ ls
temp2.txt temp.txt
ajay@testdemo:~/test$ pwd
/home/ajay/test
ajay@testdemo:~/test$ cp temp2.txt /home/ajay/Documents/
ajay@testdemo:~/test$ cd ..
ajay@testdemo:~$ cd Documents/
ajay@testdemo:~/Documents$ ls
ajaysayshi.txt  bash_script2.sh  list4.txt  script3.sh  temp2.txt
backup.sh       list2.txt        list99.txt script4.sh
bash_script1.sh list3.txt        list.txt   siem-adm
ajay@testdemo:~/Documents$
```

In the above example, we are copy the temp2.txt file from one directory to another, it will now exist in both directories.

`mv temp3.txt /home/ajay/Documents` (Please note your folders and files may be different)

Expected output:

```
ajay@testdemo:~/test$ pwd
/home/ajay/test
ajay@testdemo:~/test$ touch temp3.txt
ajay@testdemo:~/test$ ls -lh
total 0
-rw-rw-r-- 1 ajay ajay 0 Apr 17 17:19 temp2.txt
-rw-rw-r-- 1 ajay ajay 0 Apr 18 11:58 temp3.txt
-rw-rw-r-- 1 ajay ajay 0 Apr 17 17:18 temp.txt
ajay@testdemo:~/test$ mv temp3.txt /home/ajay/Documents/
ajay@testdemo:~/test$ ls -lh
total 0
-rw-rw-r-- 1 ajay ajay 0 Apr 17 17:19 temp2.txt
-rw-rw-r-- 1 ajay ajay 0 Apr 17 17:18 temp.txt
ajay@testdemo:~/test$ cd /home/ajay/Documents/
ajay@testdemo:~/Documents$ ls -lh
total 60K
-rw-rw-r-- 1 ajay ajay 0 Nov 5 11:12 ajaysayshi.txt
-rwxr-xr-x 1 ajay ajay 85 Nov 4 11:03 backup.sh
-rwxrwxr-x 1 ajay ajay 19 Nov 4 10:49 bash_script.sh
-rwxrwxr-x 1 ajay ajay 119 Nov 4 10:53 bash_script2.sh
-rw-rw-r-- 1 ajay ajay 9 Oct 29 13:39 list2.txt
-rw-rw-r-- 1 ajay ajay 84 Oct 29 13:40 list3.txt
-rw-rw-r-- 1 ajay ajay 17K Oct 29 14:01 list4.txt
-rw-rw-r-- 1 ajay ajay 93 Nov 4 10:45 list99.txt
-rw-rw-r-- 1 ajay ajay 93 Oct 29 13:37 list.txt
-rwxrwxr-x 1 ajay ajay 102 Nov 4 18:31 script3.sh
-rwxr-xr-x 1 ajay ajay 40 Nov 4 18:33 script4.sh
drwxrwxr-x 2 ajay ajay 4.0K Mar 27 12:43 siem-admin
-rw-rw-r-- 1 ajay ajay 0 Apr 17 17:20 temp2.txt
-rw-rw-r-- 1 ajay ajay 0 Apr 18 11:58 temp3.txt
ajay@testdemo:~/Documents$
```

<https://www.computerhope.com/unix/umv.htm>

The mv command moves, or renames, files and directories on your filesystem.

Syntax

Move source file(s) to a directory named destination:

`mv [options] source [source2 ...] destination`


In this exercise, we are creating a file w/ touch, then moving it from the local directory to another directory. (Please use your local file system, it may be different from this example.)

You will then list the directory, you won't see the file any more, when you navigate to the destination directory where you copied the file, you will be able to see that the file has been moved there.

`mv temp3.txt temp-changed.txt`

Expected output:

```
ajay@testdemo:~/Documents$ mv temp3.txt temp-changed.txt
ajay@testdemo:~/Documents$ ls -lh
total 60K
-rw-rw-r-- 1 ajay ajay  0 Nov  5 11:12 ajaysayshi.txt
-rwxr-xr-x 1 ajay ajay 85 Nov  4 11:03 backup.sh
-rwxrwxr-x 1 ajay ajay 19 Nov  4 10:49 bash_script1.sh
-rwxrwxr-x 1 ajay ajay 119 Nov  4 10:53 bash_script2.sh
-rw-rw-r-- 1 ajay ajay  9 Oct 29 13:39 list2.txt
-rw-rw-r-- 1 ajay ajay 84 Oct 29 13:40 list3.txt
-rw-rw-r-- 1 ajay ajay 17K Oct 29 14:01 list4.txt
-rw-rw-r-- 1 ajay ajay 93 Nov  4 10:45 list99.txt
-rw-rw-r-- 1 ajay ajay 93 Oct 29 13:37 list.txt
-rwxrwxr-x 1 ajay ajay 102 Nov  4 18:31 script3.sh
-rwxr-xr-x 1 ajay ajay 40 Nov  4 18:33 script4.sh
drwxrwxr-x 2 ajay ajay 4.0K Mar 27 12:43 siem-adm
-rw-rw-r-- 1 ajay ajay  0 Apr 17 17:20 temp2.txt
-rw-rw-r-- 1 ajay ajay  0 Apr 18 11:58 temp-changed.txt
```



Syntax

Rename a file named source to destination:

`mv [options] source destination`

In this exercise we are using the `mv` command to rename a file.

`rm temp2.txt`

Expected output:

```
ajay@testdemo:~/Documents$ ls -l
total 60
-rw-rw-r-- 1 ajay ajay    0 Nov  5 11:12 ajaysayshi.txt
-rwxr-xr-x 1 ajay ajay   85 Nov  4 11:03 backup.sh
-rwxrwxr-x 1 ajay ajay   19 Nov  4 10:49 bash_script1.sh
-rwxrwxr-x 1 ajay ajay  119 Nov  4 10:53 bash_script2.sh
-rw-rw-r-- 1 ajay ajay    9 Oct 29 13:39 list2.txt
-rw-rw-r-- 1 ajay ajay   84 Oct 29 13:40 list3.txt
-rw-rw-r-- 1 ajay ajay 16936 Oct 29 14:01 list4.txt
-rw-rw-r-- 1 ajay ajay   93 Nov  4 10:45 list99.txt
-rw-rw-r-- 1 ajay ajay   93 Oct 29 13:37 list.txt
-rwxrwxr-x 1 ajay ajay  102 Nov  4 18:31 script3.sh
-rwxr-xr-x 1 ajay ajay   40 Nov  4 18:33 script4.sh
drwxrwxr-x 2 ajay ajay  4096 Mar 27 12:43 siem-adm
-rw-rw-r-- 1 ajay ajay    0 Apr 17 17:20 temp2.txt
-rw-rw-r-- 1 ajay ajay    0 Apr 18 11:58 temp-changed.txt
ajay@testdemo:~/Documents$ rm temp2.txt
ajay@testdemo:~/Documents$ ls -l
total 60
-rw-rw-r-- 1 ajay ajay    0 Nov  5 11:12 ajaysayshi.txt
-rwxr-xr-x 1 ajay ajay   85 Nov  4 11:03 backup.sh
-rwxrwxr-x 1 ajay ajay   19 Nov  4 10:49 bash_script1.sh
-rwxrwxr-x 1 ajay ajay  119 Nov  4 10:53 bash_script2.sh
-rw-rw-r-- 1 ajay ajay    9 Oct 29 13:39 list2.txt
-rw-rw-r-- 1 ajay ajay   84 Oct 29 13:40 list3.txt
-rw-rw-r-- 1 ajay ajay 16936 Oct 29 14:01 list4.txt
-rw-rw-r-- 1 ajay ajay   93 Nov  4 10:45 list99.txt
-rw-rw-r-- 1 ajay ajay   93 Oct 29 13:37 list.txt
-rwxrwxr-x 1 ajay ajay  102 Nov  4 18:31 script3.sh
-rwxr-xr-x 1 ajay ajay   40 Nov  4 18:33 script4.sh
drwxrwxr-x 2 ajay ajay  4096 Mar 27 12:43 siem-adm
-rw-rw-r-- 1 ajay ajay    0 Apr 18 11:58 temp-changed.txt
```

Now its deleted.

<https://www.computerhope.com/unix/urm.htm>

The **rm** ("remove") command is used to delete files. When used recursively, it may be used to delete directories.

rm removes each file specified on the command line. By default, it does not remove directories.

The removal process unlinks a file name in a filesystem from its associated data, and marks that space on the storage device as usable by future writes. In other words, when you remove a file, the data in the file isn't changed, but it's no longer associated with a filename.

The data itself is not destroyed, but after being unlinked with **rm**, it becomes inaccessible. Remove your files wisely! It's not like putting something in the Windows Recycle Bin; once you **rm** a file or directory, there is no way to undo it.

PROTIP: If you want to completely wipe the data on the disk, use the **shred** command instead. **shred** will overwrite the file's contents so that they cannot be reconstructed later.

ls

ls test

rm test (This is DESIGNED to FAIL on purpose to illustrate the power of the -r argument)

rm -r test

ls

```
ajay@testdemo:~$ ls
Backup  directory.txt  examples.desktop  Pictures  test
d2      Documents      Music             Public    test2.tar
Desktop Downloads      permissions       Templates Videos
ajay@testdemo:~$ ls test
temp2.txt  temp.txt
ajay@testdemo:~$ rm test
rm: cannot remove 'test': Is a directory
ajay@testdemo:~$ rm -r test
ajay@testdemo:~$ ls
Backup  directory.txt  examples.desktop  Pictures  test2.tar
d2      Documents      Music             Public    Videos
Desktop Downloads      permissions       Templates
ajay@testdemo:~$
```

Removing directories

By default, rm does not remove directories. If the -r/-R/--recursive option is specified, however, rm will remove any matching directories and their contents.

Notice that to remove a directory with items in it, you have to use the -r argument.

Be CAREFUL, you will delete the ENTIRE directory substructure, pay attention and understand what is contained in the folder that you plan to delete.

env

Expected output:

```
ajay@testdemo:~$ env
XDG_VTNR=7
XDG_SESSION_ID=c2
XDG_GREETER_DATA_DIR=/var/lib/lightdm-data/ajay
CLUTTER_IM_MODULE=xim
SESSION=ubuntu
GPG_AGENT_INFO=/home/ajay/.gnupg/S.gpg-agent:0:1
TERM=xterm-256color
VTE_VERSION=4205
XDG_MENU_PREFIX=gnome-
SHELL=/bin/bash
QT_LINUX_ACCESSIBILITY_ALWAYS_ON=1
WINDOWID=52428810
UPSTART_SESSION=unix:abstract=/com/ubuntu/upstart-session
GNOME_KEYRING_CONTROL=
GTK_MODULES=gail:atk-bridge:unity-gtk-module
USER=ajay
LS_COLORS=rs=0:di=01;34:ln=01;36:mh=00:pi=40;33:so=01;35:
0;33:01;or=40;31:01;mi=00:su=37;41:sg=30;43:ca=30;41:tw=3
=01;32:*.tar=01;31:*.tgz=01;31:*.arc=01;31:*.arj=01;31:*.
z4=01;31:*.lzh=01;31:*.lzma=01;31:*.tlz=01;31:*.txz=01;31
*.zip=01;31:*.z=01;31:*.Z=01;31:*.dz=01;31:*.gz=01;31:*.l
=01;31:*.xz=01;31:*.bz2=01;31:*.bz=01;31:*.tbz=01;31:*.tb
=01;31:*.rpm=01;31:*.jar=01;31:*.war=01;31:*.ear=01;31:*.
lz=01;31:*.ace=01;31:*.zoo=01;31:*.cpio=01;31:*.7z=01;31:
jpg=01;35:*.jpeg=01;35:*.gif=01;35:*.bmp=01;35:*.pbm=01;3
:*.tga=01;35:*.xbm=01;35:*.xpm=01;35:*.tif=01;35:*.tiff=0
;35:*.svgz=01;35:*.mng=01;35:*.pcx=01;35:*.mov=01;35:*.mp
v=01;35:*.mkv=01;35:*.webm=01;35:*.ogm=01;35:*.mp4=01;35:
*.vob=01;35:*.qt=01;35:*.nuv=01;35:*.wmv=01;35:*.asf=01;3
:*.flc=01;35:*.avi=01;35:*.fli=01;35:*.flv=01;35:*.gl=01;
:*.xwd=01;35:*.yuv=01;35:*.cgm=01;35:*.emf=01;35:*.ogv=01
36:*.au=00;36:*.flac=00;36:*.m4a=00;36:*.mid=00;36:*.midi
00;36:*.mpc=00;36:*.ogg=00;36:*.ra=00;36:*.wav=00;36:*.og
x=00;36:*.xspf=00;36:
QT_ACCESSIBILITY=1
XDG_SESSION_PATH=/org/freedesktop/DisplayManager/Session0
UNITY_HAS_3D_SUPPORT=false
XDG_SEAT_PATH=/org/freedesktop/DisplayManager/Seat0
SSH_AUTH_SOCK=/run/user/1000/keyring/ssh
DEFAULTS_PATH=/usr/share/gconf/ubuntu.default.path
XDG_CONFIG_DIRS=/etc/xdg/xdg-ubuntu:/usr/share/upstart/xd
UNITY_DEFAULT_PROFILE=unity-lowgfx
```

<https://www.computerhope.com/unix/uenv.htm>

env is a shell command for Linux, Unix, and Unix-like operating systems. It can be used to print a list of the current environment variables, or to run another program in a custom environment without modifying the current one.

Description

If **env** is run without any options, it prints the variables of the current environment.

WORKING WITH FILES

LET'S DO WAR AND PEACE

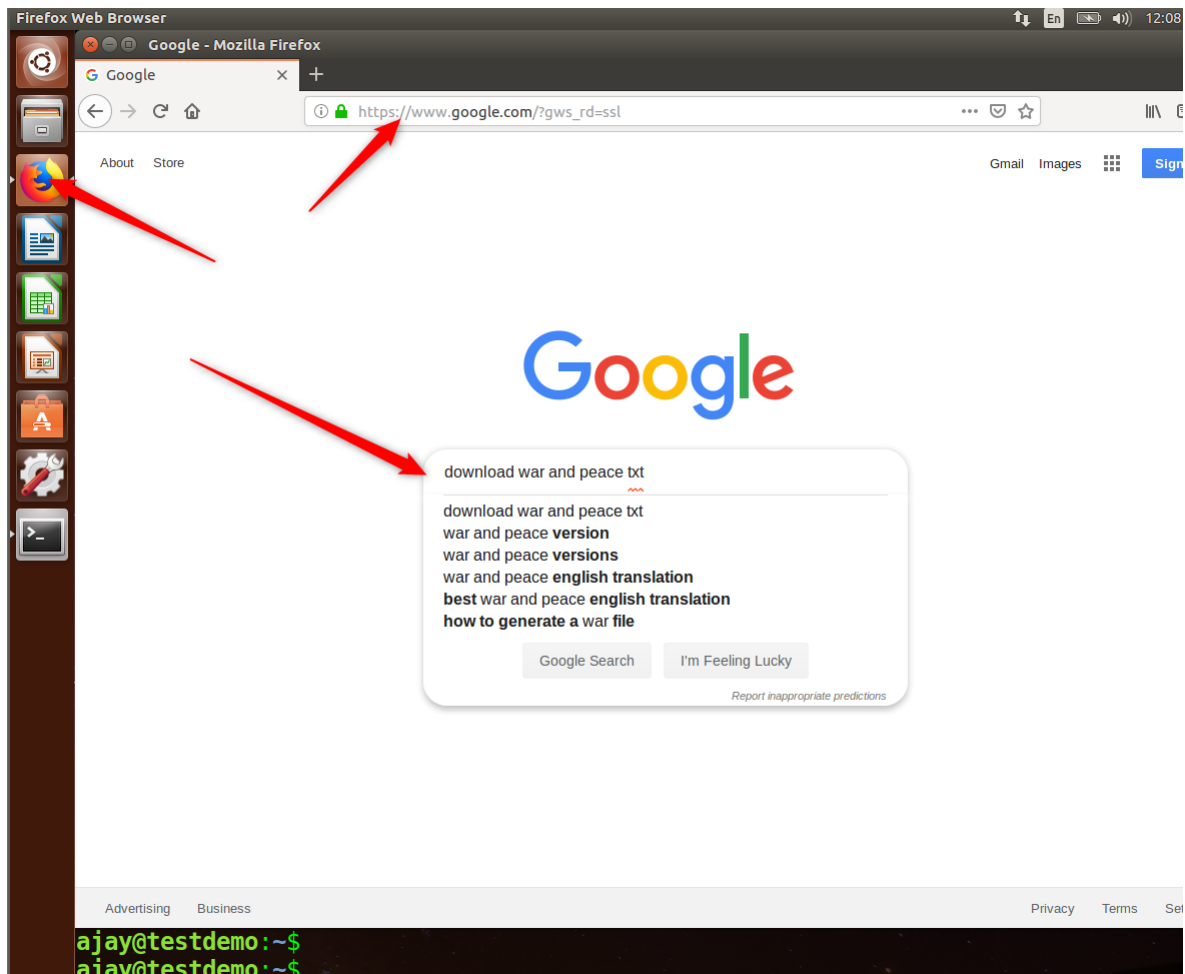
Here are several commands and programs provided by Linux for viewing the contents of file. Working with files is one of the daunting task, most of the computer users be it newbie, regular user, advanced user, developer, admin, etc performs. Working with files effectively and efficiently is an art.

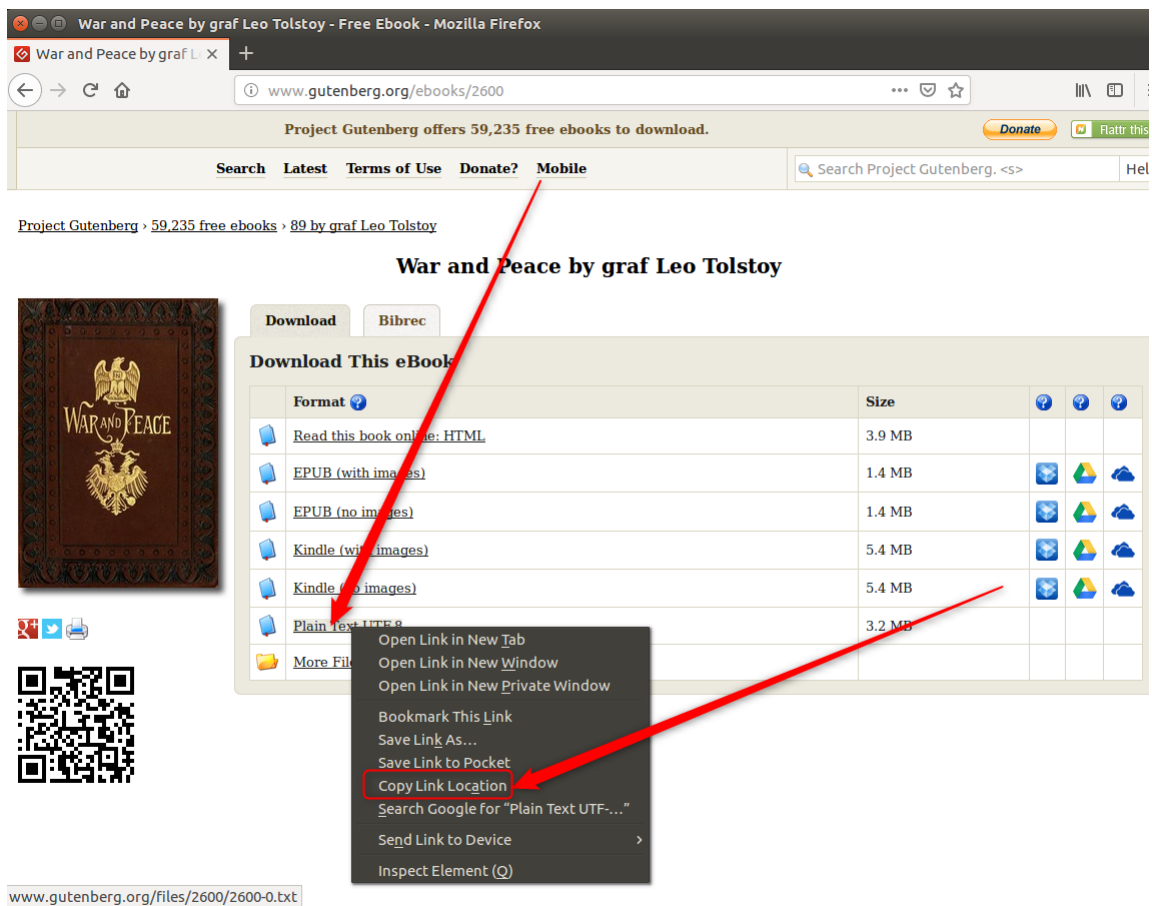
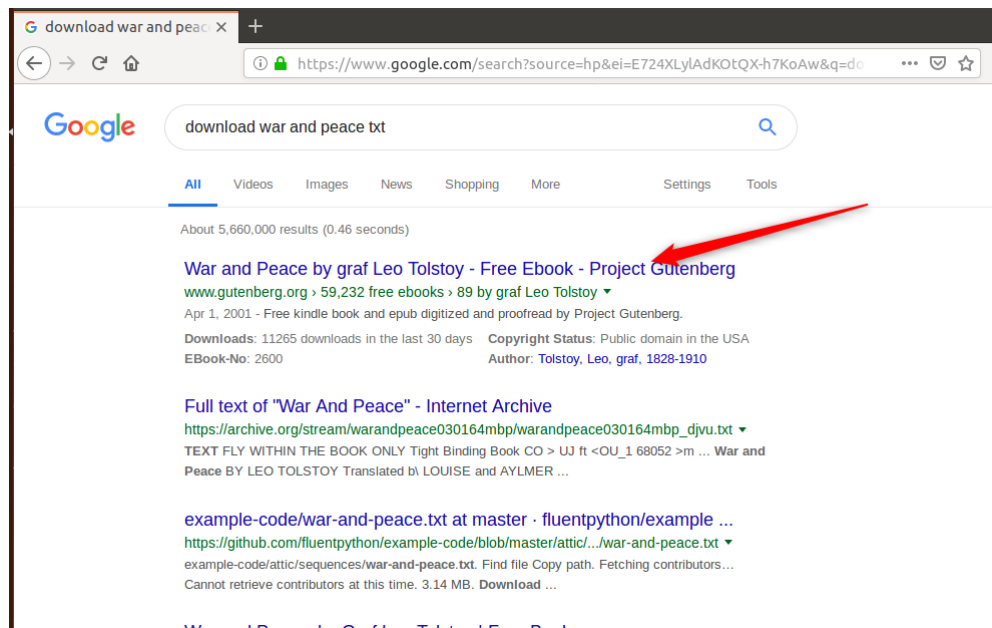
<https://www.tecmint.com/view-contents-of-file-in-linux/>

We will start by downloading a copy of war and peace as a text file from the below link:

<http://www.gutenberg.org/files/2600/2600-0.txt>

Step by step pictorial instructions:





```
ajay@testdemo:~$  
ajay@testdemo:~$  
ajay@testdemo:~$  
ajay@testdemo:~$ ls  
Backup  directory.txt  exam  Files  Pictures  test2.tar  
d2      Documents      Mus.  Read-Only  Public  Videos  
Desktop Downloads    per.  ✓ Show Menubar  Templates  
ajay@testdemo:~$ mkdir test  
ajay@testdemo:~$ cd test  
ajay@testdemo:~/test$ wget
```

```
ajay@testdemo:~$ mkdir test  
ajay@testdemo:~$ cd test  
ajay@testdemo:~/test$ wget http://www.gutenberg.org/files/2600/2600-0.txt
```

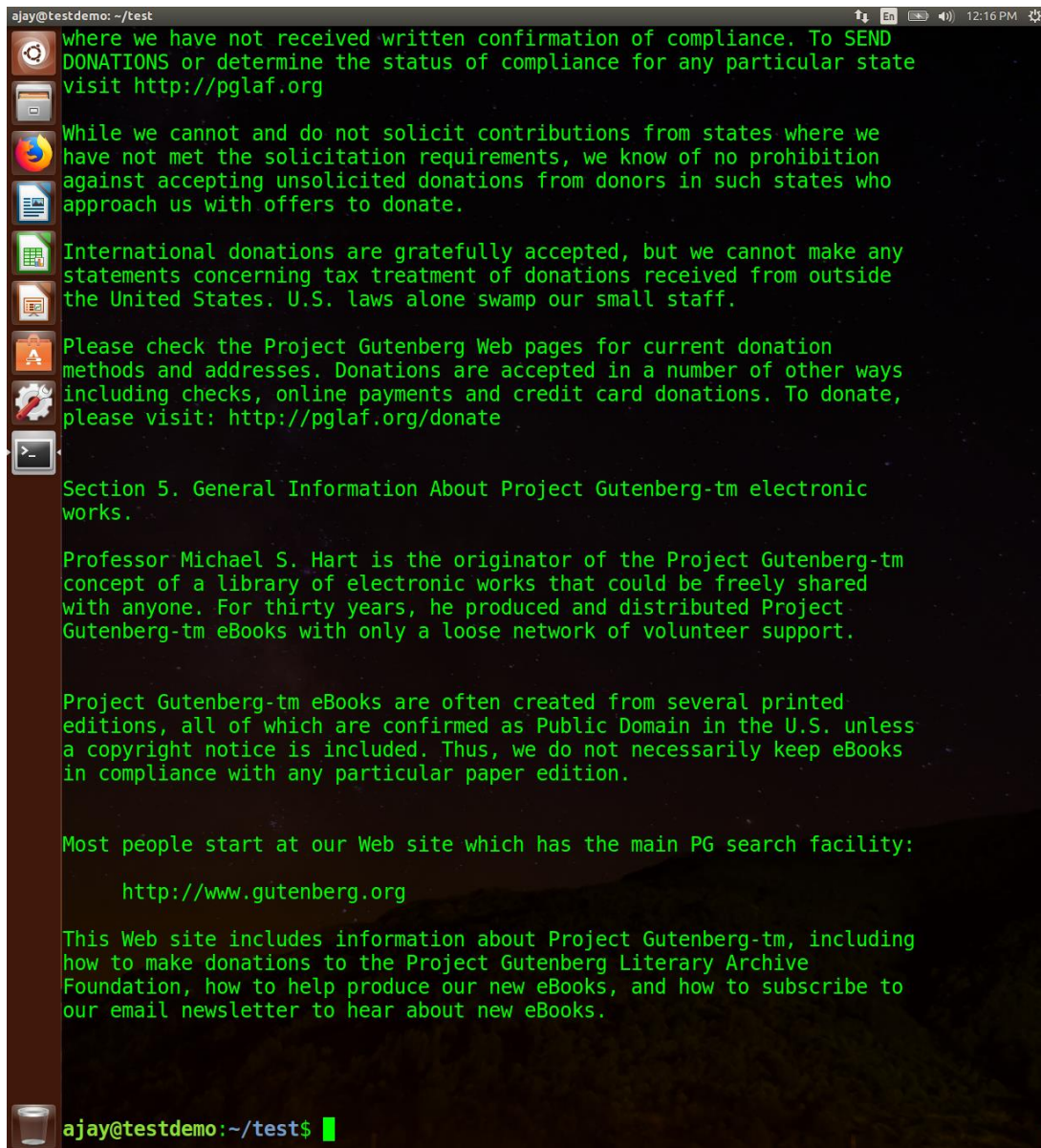
ls -l

```
ajay@testdemo:~/test$ wget http://www.gutenberg.org/files/2600/2600-0.txt  
--2019-04-18 12:12:29-- http://www.gutenberg.org/files/2600/2600-0.txt  
Resolving www.gutenberg.org (www.gutenberg.org)... 152.19.134.47, 2610:28:3090:300  
0:0:bad:cafe:47  
Connecting to www.gutenberg.org (www.gutenberg.org)|152.19.134.47|:80... connected  
.  
HTTP request sent, awaiting response... 200 OK  
Length: 3359545 (3.2M) [text/plain]  
Saving to: '2600-0.txt'  
  
2600-0.txt          100%[=====>] 3.20M 2.55MB/s  in 1.3s  
2019-04-18 12:12:30 (2.55 MB/s) - '2600-0.txt' saved [3359545/3359545]  
  
ajay@testdemo:~/test$ ls -lh  
total 3.3M  
-rw-rw-r-- 1 ajay ajay 3.3M Jan 23 05:42 2600-0.txt  
ajay@testdemo:~/test$ mv 2600-0.txt warandpeace.txt  
ajay@testdemo:~/test$ ls -lh  
total 3.3M  
-rw-rw-r-- 1 ajay ajay 3.3M Jan 23 05:42 warandpeace.txt
```

The file will download as 2600-0.txt NOT very informative. Using the skills we just learned, let us rename the file by using the mv command.

cat warandpeace.txt

Expected output:

A terminal window titled 'ajay@testdemo: ~/test' with a dark background and green text. The output of the 'cat warandpeace.txt' command is displayed. The text is a donation request from Project Gutenberg, mentioning compliance requirements, solicitation rules, and donation methods. It also includes a section titled 'Section 5. General Information About Project Gutenberg-tm electronic works.' which describes the originator, Michael S. Hart, and the project's goals. The terminal window has a sidebar with various application icons on the left and a system status bar at the top right showing '12:16 PM' and other icons. The prompt 'ajay@testdemo:~/test\$' is visible at the bottom.

```
ajay@testdemo: ~/test
where we have not received written confirmation of compliance. To SEND
DONATIONS or determine the status of compliance for any particular state
visit http://pglaf.org

While we cannot and do not solicit contributions from states where we
have not met the solicitation requirements, we know of no prohibition
against accepting unsolicited donations from donors in such states who
approach us with offers to donate.

International donations are gratefully accepted, but we cannot make any
statements concerning tax treatment of donations received from outside
the United States. U.S. laws alone swamp our small staff.

Please check the Project Gutenberg Web pages for current donation
methods and addresses. Donations are accepted in a number of other ways
including checks, online payments and credit card donations. To donate,
please visit: http://pglaf.org/donate

Section 5. General Information About Project Gutenberg-tm electronic
works.

Professor Michael S. Hart is the originator of the Project Gutenberg-tm
concept of a library of electronic works that could be freely shared
with anyone. For thirty years, he produced and distributed Project
Gutenberg-tm eBooks with only a loose network of volunteer support.

Project Gutenberg-tm eBooks are often created from several printed
editions, all of which are confirmed as Public Domain in the U.S. unless
a copyright notice is included. Thus, we do not necessarily keep eBooks
in compliance with any particular paper edition.

Most people start at our Web site which has the main PG search facility:

    http://www.gutenberg.org

This Web site includes information about Project Gutenberg-tm, including
how to make donations to the Project Gutenberg Literary Archive
Foundation, how to help produce our new eBooks, and how to subscribe to
our email newsletter to hear about new eBooks.

ajay@testdemo:~/test$
```

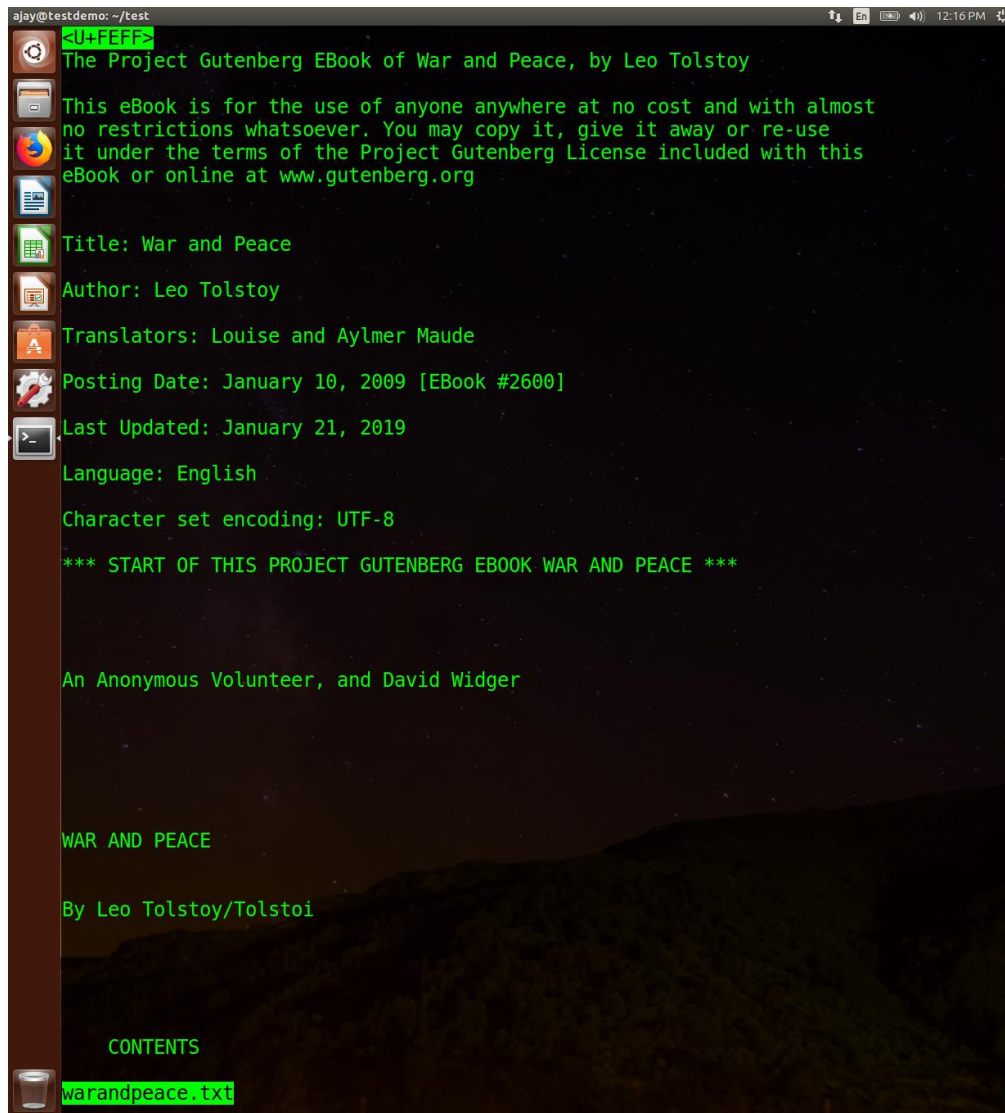
<https://www.computerhope.com/unix/ucatl.htm>

cat stands for "catenate." It reads data from files, and outputs their contents. It is the simplest way to display the contents of a file at the command line.

You will see it BLOWS through the entire contents of War and Peace. Good, it works, but for such a long file cat isn't that useable.

Less warandpeace.txt

Expected output:



```
eJay@testdemo: ~/test
<U+FEFF>
The Project Gutenberg EBook of War and Peace, by Leo Tolstoy

This eBook is for the use of anyone anywhere at no cost and with almost
no restrictions whatsoever. You may copy it, give it away or re-use
it under the terms of the Project Gutenberg License included with this
eBook or online at www.gutenberg.org

Title: War and Peace
Author: Leo Tolstoy
Translators: Louise and Aylmer Maude
Posting Date: January 10, 2009 [EBook #2600]
Last Updated: January 21, 2019
Language: English
Character set encoding: UTF-8

*** START OF THIS PROJECT GUTENBERG EBOOK WAR AND PEACE ***

An Anonymous Volunteer, and David Widger

WAR AND PEACE

By Leo Tolstoy/Tolstoi

CONTENTS
warandpeace.txt
```

When you are ready to exit type **q**.

<https://www.computerhope.com/unix/less.htm>

<https://www.thegeekstuff.com/2010/02/unix-less-command-10-tips-for-effective-navigation/>

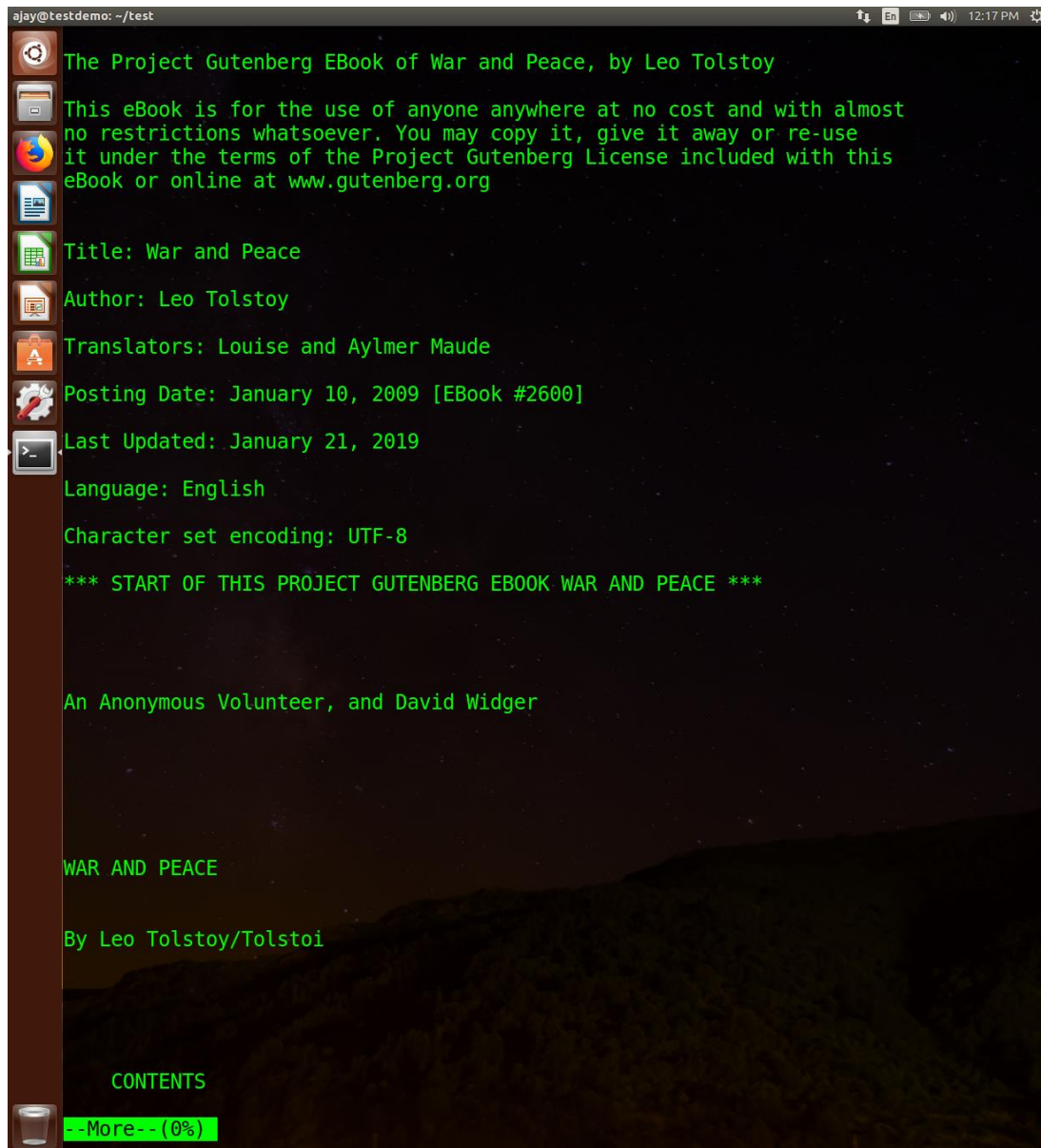
less is a simple, feature-rich command-line file viewer.

Description

less is a program similar to **more**, but it has many more features. **less** does not have to read the entire input file before starting, so with large input files it starts up faster than text editors like **vi**.

More warandpeace.txt

Expected output:

A terminal window titled 'ajay@testdemo: ~/test' showing the output of the 'more' command. The text is displayed in green on a dark background. The output includes the title 'The Project Gutenberg EBook of War and Peace, by Leo Tolstoy', a copyright notice, metadata (Title, Author, Translators, Posting Date, Last Updated, Language, Character set encoding), and the start of the book's content. The terminal shows the 'more' command's pagination interface with a status bar at the bottom indicating '--More-- (0%)'.

```
ajay@testdemo: ~/test
The Project Gutenberg EBook of War and Peace, by Leo Tolstoy

This eBook is for the use of anyone anywhere at no cost and with almost
no restrictions whatsoever. You may copy it, give it away or re-use
it under the terms of the Project Gutenberg License included with this
eBook or online at www.gutenberg.org

Title: War and Peace
Author: Leo Tolstoy
Translators: Louise and Aylmer Maude
Posting Date: January 10, 2009 [EBook #2600]
Last Updated: January 21, 2019
Language: English
Character set encoding: UTF-8

*** START OF THIS PROJECT GUTENBERG EBOOK WAR AND PEACE ***

An Anonymous Volunteer, and David Widger

WAR AND PEACE

By Leo Tolstoy/Tolstoi

CONTENTS

--More-- (0%)
```

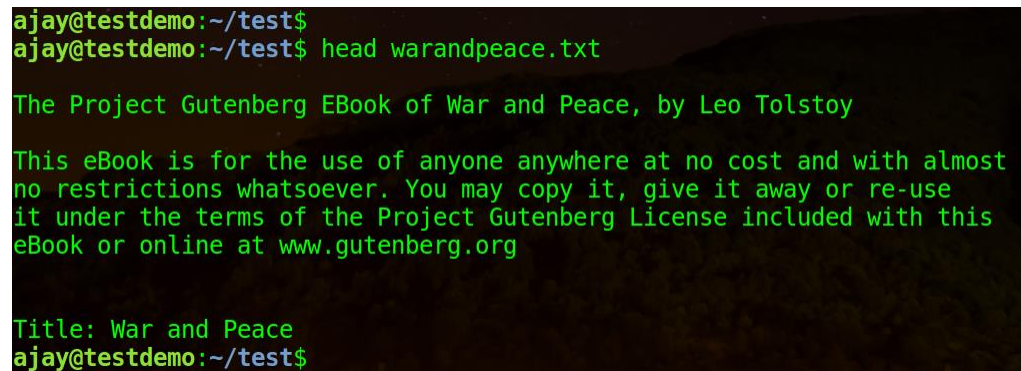
<https://www.tecmint.com/linux-more-command-and-less-command-examples/>

more is a *nix command line used to display the contents of a file in a console.

In order to navigate through the file line by line press Enter key or press Spacebar key to navigate one page at a time, the page being your current terminal screen size. To exit the command just press **q** key.

head warandpeace.txt

Expected output:

A terminal window with a dark background and green text. The prompt is 'ajay@testdemo:~/test\$'. The command 'head warandpeace.txt' has been entered. The output shows the first ten lines of the file: 'The Project Gutenberg EBook of War and Peace, by Leo Tolstoy', 'This eBook is for the use of anyone anywhere at no cost and with almost no restrictions whatsoever. You may copy it, give it away or re-use it under the terms of the Project Gutenberg License included with this eBook or online at www.gutenberg.org', and 'Title: War and Peace'. The prompt 'ajay@testdemo:~/test\$' is visible again at the bottom.

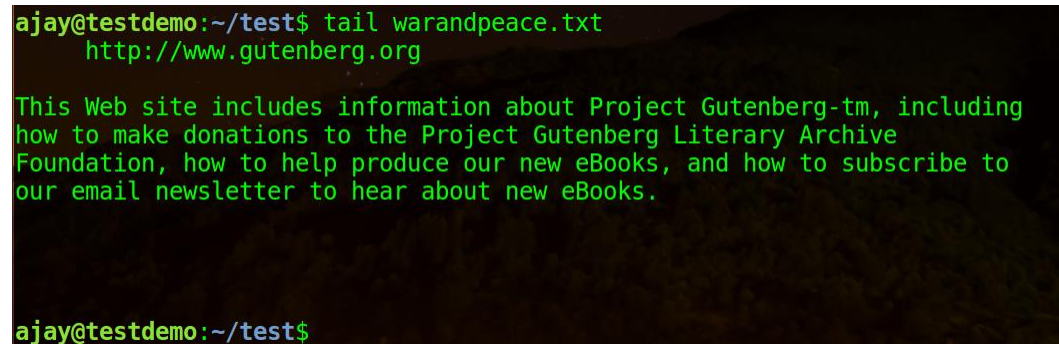
```
ajay@testdemo:~/test$ head warandpeace.txt
The Project Gutenberg EBook of War and Peace, by Leo Tolstoy
This eBook is for the use of anyone anywhere at no cost and with almost
no restrictions whatsoever. You may copy it, give it away or re-use
it under the terms of the Project Gutenberg License included with this
eBook or online at www.gutenberg.org
Title: War and Peace
ajay@testdemo:~/test$
```

The **head** command reads the first ten lines of an any given file name. The basic syntax of head command is:

head [options] [file(s)]

tail warandpeace.txt

Expected output:

A terminal window with a dark background and green text. The prompt is 'ajay@testdemo:~/test\$'. The command 'tail warandpeace.txt' has been entered. The output shows the last ten lines of the file: 'http://www.gutenberg.org', 'This Web site includes information about Project Gutenberg-tm, including how to make donations to the Project Gutenberg Literary Archive Foundation, how to help produce our new eBooks, and how to subscribe to our email newsletter to hear about new eBooks.', and 'ajay@testdemo:~/test\$' at the bottom.

```
ajay@testdemo:~/test$ tail warandpeace.txt
http://www.gutenberg.org
This Web site includes information about Project Gutenberg-tm, including
how to make donations to the Project Gutenberg Literary Archive
Foundation, how to help produce our new eBooks, and how to subscribe to
our email newsletter to hear about new eBooks.
ajay@testdemo:~/test$
```

The **tail** command allows you to display last ten lines of any text file. Similar to the head command above, **tail** command also support options 'n' number of lines and 'n' number of characters.

The basic syntax of tail command is:

tail [options] [filenames]

APPENDIX:

MAIN DIRECTORIES

/bin is a place for most commonly used terminal commands, like ls, mount, rm, etc.

/boot contains files needed to start up the system, including the Linux kernel, a RAM disk image and bootloader configuration files.

/dev contains all device files, which are not regular files but instead refer to various hardware devices on the system, including hard drives.

/etc contains system-global configuration files, which affect the system's behavior for all users. – How its pronounced.

/home home sweet home, this is the place for users' home directories.

/lib contains very important dynamic libraries and kernel modules

/media is intended as a mount point for external devices, such as hard drives or removable media (floppies, CDs, DVDs).

/mnt is also a place for mount points, but dedicated specifically to "temporarily mounted" devices, such as network filesystems.

/opt can be used to store additional software for your system, which is not handled by the package manager.

/proc is a virtual filesystem that provides a mechanism for kernel to send information to processes.

/root is the superuser's home directory, not in /home/ to allow for booting the system even if /home/ is not available.

/run is a tmpfs (temporary file system) available early in the boot process where ephemeral run-time data is stored. Files under this directory are removed or truncated at the beginning of the boot process.

(It deprecates various legacy locations such as `/var/run`, `/var/lock`, `/lib/init/rw` in otherwise non-ephemeral directory trees as well as `/dev/*` and `/dev/shm` which are not device files.)

/sbin contains important administrative commands that should generally only be employed by the superuser.

/srv can contain data directories of services such as HTTP (`/srv/www/`) or FTP.

/sys is a virtual filesystem that can be accessed to set or obtain information about the kernel's view of the system.

/tmp is a place for temporary files used by applications.

/usr contains the majority of user utilities and applications, and partly replicates the root directory structure, containing for instance, among others, `/usr/bin/` and `/usr/lib`.

/var is dedicated to variable data, such as logs, databases, websites, and temporary spool (e-mail etc.) files that persist from one boot to the next. A notable directory it contains is `/var/log` where system log files are kept.

ABSOLUTE PATH

An absolute path name, pointing to what is normally an executable file on an Ubuntu system:

`/usr/bin/test`

An absolute path name, but pointing to a directory instead of a regular file:

`/usr/bin/`

A relative path name, which will point to /usr/bin/test only if the current directory is /usr/:

bin/test

A relative path name, which will point to /usr/bin/test if the current directory is any directory in /usr/, for instance /usr/share/:

../bin/test

A path name using the special shortcut ~, which refers to the current user's home directory:

~/Desktop/

Path names can contain almost any character, but some characters, such as space, must be escaped in most software, usually by enclosing the name in quotation marks:

"~/Examples/Experience ubuntu.ogg"

or by employing the escape character \:

~/Examples/Experience\ ubuntu.ogg

./config means you're calling something in the current working directory. In this case config is an executable. You have to specify the path for executables if they're outside your \$PATH variable and that's why config isn't enough.

../config would be used if the config executable were in the parent of the current working directory.

CLI KNOWLEDGE:

1. root@linux1:/# If you are logged in as root or a superuser instead of \$ you will get a #

1. The root user has considerable power, so use it with caution. When you have root privileges, most prompts include a trailing pound sign (#). Ordinary user privileges are usually delineated by a different character, commonly a dollar sign (\$). Your actual prompt may look different than the examples in this tutorial. Your prompt may include your user name,

hostname, current directory, date, or time that the prompt was printed, and so on. We will discuss root and superusers shortly.

2. The shell's main function is to interpret your commands so you can interact with your Linux system.
3. If a line contains a **#** character, then all remaining characters on the line are ignored. So, a **#** character may indicate a comment as well as a root prompt. Which it is should be evident from the context.
4. You need to quote strings, using either double quotes (") or single quotes (').
 2. a **string** is any finite sequence of characters (i.e., letters, numerals, symbols and punctuation marks).
 3. Bash uses white space, such as blanks, tabs, and new line characters, to separate your input line into tokens, which are then passed to your command.
 4. Quoting strings preserves additional white space and makes the whole string a single token.

CLI NAVIGATION:

echo

The echo program displays text. It's a handy way to create customized output in your terminal.

echo is a fundamental command found in most operating systems. It is frequently used in scripts, batch files, and as part of individual commands; anywhere you may need to output text.

echo [SHORT-OPTION]... [STRING]...

pwd (print working directory) – WHERE THE HECK AM I? =)

Prints the current working directory on your screen. This is the directory where you are currently located. When you manipulate files and sub-directories, this is where they will (by default) be.

Here is an example of using pwd:

pwd

/homeb/bpowell

cd (change directory)

This command is used to change the current working directory.

Here are some examples of using cd:

cd datafiles cd .. cd / cd \$HOME

The last example shows the use of a UNIX environment variable. The \$HOME variable always contains the location of your HOME directory. (We will discuss BASH variables later in more detail)

When an environment variable is used in a command, the contents of the variable and substituted for its name, so the above example would end up doing the same thing as if you typed in:

cd /homeb/bpowell

a. **env**

- i. You can print all of your environment variables by running the "env" command.

b. **mkdir** (make directory)

- i. This command makes a sub-directory under the current working directory.
- ii. **mkdir junk**

c. **rmdir** (remove directory)

- i. This command removes (deletes) a sub-directory under the current working directory. The directory to be removed must be empty of all files and sub-directories.
- ii. **rmdir junk**

d. **touch**

- i. **touch** changes file timestamps. It is also an easy way to create empty files.

e. **date**

- i. **date** will display the time and date as well as time zone information.

a. **Manipulate files**

- i. **ls** (list files)

1. This command is similar to DIR in DOS; it displays a list of all the files in the directory. New users don't have any files in their home directory.
2. Here is an example of using this command;
3. **ls**
4. You will note that the shell prompt reappears, and there is no file listing of the directory contents. This does not mean that there are no files stored there. Just like DOS, UNIX systems support hidden files.
5. Here is an example of using the command with switches:
 - a. **ls -la**
6. The switch **l** stands for long listing, **and the a switch is for all files, including directories and hidden files.**
7. UNIX responds with the following listing of the directory
 - a. **ls -la**
 - b. total 6 -rw-rw-r-- 1 jmsmith staff 526 Apr 15
11:03 myletter

- ii. **less** (page through a text file)

1. This command allows you to view a text file without fear of accidentally modifying it, as you would with a text editor.
2. the key **q** allows you to exit. Space bar to go from page to page.
3. Example:
 - a. **less /etc/hosts**

- iii. **cat** (concatenate files)

1. This command is used to combine files or to print files to the screen. By default, the cat command sends its output to your screen (in UNIX we call this standard-output or stdout for short).
2. The following command can be used to view the file, ".profile" on stdout:

3. `cat .profile`
 4. The format of this command specifies that `cat` will use the file, ".profile" as its input, and send the output to your screen.
 5. Note: if the file is very large or is not a plain text file, `cat` will try to print it to the screen anyway, sometimes with undesirable results. The `less` command is better suited for viewing files than the `cat` command.
- i. **cp** (copy files)
 - i. This command stands for copy, and is used for copying one file to another.
 - ii. Example:
 - i. `cp .profile temp2`
 - iii. This copies the file .profile to another called temp2. If temp2 had already existed, its previous contents would've been erased.
 - iv. Files can also be copied to another directory. The command
 - v. `cp * /usr/tmp`
 - i. would copy all the files in the current directory to the directory /usr/tmp.
 - ii. **mv** (move files)
 - i. The `mv` command is used for moving or renaming files.
 - ii. Example:
 - i. `mv temp temp2`
 - iii. This renames the file temp to temp2.
 - iv. As an example (do not type this), the command:
 - v. `mv temp2 /tmp`
 - vi. would move the file temp2 into the directory /tmp (it would no longer appear in your home directory).
 - i. **rm** (remove files)
 1. The `rm` utility is used for erasing files and directories.
 - a. Example:
 - i. `rm temp2`
 2. This removes the file. Once a file is removed, it cannot be restored. To cover situations where mistakes might occur, a switch `-i` appended to this command will request a Yes or No response before deleting the file.
 3. Example:

a. `rm -i temp1`

4. NOTE that switches are written before the filenames. Answer Y to the prompt so that temp1 is removed.

ii. **head**

- i. This is used to view the first few lines of a file. It accepts a switch specifying the number of lines to view. The command

ii. `head -2 temp`

- iii. would list the first 2 lines of the file temp on your screen.

iii. **tail**

- i. This is used to view the last few lines of a file. It accepts a switch specifying the number of lines to view. The command

ii. `tail -2 temp`

iii. `tail -f logfile.log`

- iv. This forces the tail to follow the log, so it shows items being written to that file in real-time.