

# Basic Linux Tutorial

Version - 0.2  
Date - 7-March-2024

## Remotely connecting to Linux Command Line Interface (CLI) from Windows

1. You may download either of these ssh clients:-

PuTTY from <http://www.chiark.greenend.org.uk/~sgtatham/putty/>

MobaXterm:- <https://mobaxterm.mobatek.net/download-home-edition.html>

2. Once the SSH Connection is open, you should see a terminal prompt asking for your username:

**login as:**

Connect to ASPIRE2a login node via respective stakeholders (NUS/NTU/ASTAR etc..) VPN.

3. Next, enter your password. Please note that you will NOT see your cursor moving, or any characters typed (such as \*\*\*\*\*), when typing your password. This is a standard PuTTY security feature. Hit enter.

**Using keyboard-interactive authentication.**

**Password:**

4. You are now logged into your server with SSH. You should see output like this:

**example.com@n11:~\$**

## Command Shells:

A shell is a command interpreter which allows you to interact with the computer. The behavior of the command line interface will differ slightly depending on the shell program that is being used.

Depending on the shell used, some extra behaviors can be quite nifty. In this tutorial we will focus on Bash, since it is the most widely used and also one of the most powerful shells out there.

You can find out what shell you are using by the command:

**\$ echo \$SHELL**

Of course you can create a file with a list of shell commands and execute it like a program to perform a task. This is called a shell script. This is in fact the primary purpose of most shells, not the interactive command line behavior. This can be done by inserting the following line at the very top of your script.

```
#!/bin/bash
```

## Linux Directories

File and directory paths in LINUX use the forward slash "/" to separate directory names in a path.

examples:

/	"root" directory
/home/myuserid	your HOME directory
/usr	directory usr (sub-directory of / "root" directory)
/usr/STRIM100	STRIM100 is a subdirectory of /usr

## Moving around the file system

<code>pwd</code>	Show the "present working directory", or current directory.
<code>cd</code>	Change the current directory to your HOME directory.
<code>cd /usr/STRIM100</code>	Change the current directory to /usr/STRIM100.
<code>cd INIT</code>	Change current directory to INIT which is a sub-directory of the current directory.
<code>cd ..</code>	Change current directory to the parent directory of the current directory.
<code>cd \$STRMWORK</code>	Change current directory to the directory defined by the environment variable 'STRMWORK'.
<code>cd ~bob</code>	Change the current directory to the user bob's home directory (if you have permission).

## Listing directory contents:

<code>ls</code>	list a directory	<code>ls -l</code>	list a directory in long ( detailed ) format
-----------------	------------------	--------------------	----------------------------------------------

for example:

```
$ ls -l
drwxr-xr-x  4 cliff  user      1024 Jun 18 09:40
WAITRON_EARNINGS
-rw-r--r--  1 cliff  user    767392 Jun  6 14:28 scanlib.tar.gz
^  ^  ^  ^      ^  ^      ^
|  |  |  |      |  |      |
|  |  |  |      | owner  group  size  date  time  name
```

| | | |        number of links to file or directory contents  
| | |        permissions for world  
| |        permissions for members of group  
|        permissions for owner of file: r = read, w = write, x = execute --no  
permission  
type of file: - = normal file, d=directory, l = symbolic link, and  
others...

**ls -a**        List the current directory including hidden files. Hidden files start  
with "."

**ls -ld \***     List all the file and directory names in the current directory using long format. Without  
the "d" option, ls would list the contents of any sub-directory of the current. With the  
"d" option, ls just lists them like regular files.

## Changing file permissions and attributes:

**chmod 755 filename**     Changes the permissions of file to be rwx for the owner, and rx for  
the group and the world. (7 = rwx = 111 binary. 5 = r-x = 101 binary)

**chgrp user file**        Makes files belong to the group user.

**chown cliff file**        Makes cliff the owner of the file.

**chown -R cliff dir**     Makes cliff the owner of dir and everything in its directory tree.

You must be the owner of the file/directory or be root before you can do any of these things.

## Moving, renaming, and copying files

**cp file1 file2**        copy a file

**mv file1 newname**     move or rename a file **mv file1 ~/AAA/**     move file1  
into sub-directory AAA in your home directory. **rm file1 [file2 ...]**     remove or  
delete a file **rm -r dir1 [dir2...]**     recursively remove a directory and its contents  
BE CAREFUL!

**mkdir dir1 [dir2...]**     create directories

**mkdir -p dirpath**        create the directory dirpath, including all implied directories in the path.

**rmdir dir1 [dir2...]**     remove an empty directory

## Viewing and editing files

**cat filename**        Dump a file to the screen in ascii.

**more filename**        Progressively dump a file to the screen: ENTER = one line down  
SPACEBAR = page down q=quit

**less filename**        Like more, but you can use Page-Up too. Not on all systems.

**vi filename**        form. Edit a file using the vi editor. All LINUX systems will have vi in some

<b>emacs filename</b>	Edit a file using the emacs editor. Not all systems will have emacs.
<b>head filename</b>	Show the first few lines of a file.
<b>head -n filename</b>	Show the first n lines of a file.
<b>tail filename</b>	Show the last few lines of a file.
<b>tail -n filename</b>	Show the last n lines of a file.

## Environment variables

You can teach your shell to remember things for later using environment variables. For example under the bash shell:

<b>export CASROOT=/usr/local/CAS3.0</b>	Defines the variable CASROOT with the value /usr/local/CAS3.0.
<b>export LD_LIBRARY_PATH=\$CASROOT/Linux/lib</b>	Defines the variable LD_LIBRARY_PATH with the value of CASROOT with /Linux/lib appended, or /usr/local/CAS3.0/Linux/lib

By prefixing \$ to the variable name, you can evaluate it in any command:

<b>cd \$CASROOT</b>	Changes your present working directory to the value of CASROOT
<b>echo \$CASROOT</b>	Prints out the value of CASROOT, or /usr/local/CAS3.0
<b>printenv CASROOT</b>	Does the same thing in bash and some other shells.

## Interactive History

A feature of bash and tcsh (and sometimes others) you can use the up-arrow keys to access your previous commands, edit them, and re-execute them.

## Filename and Command Completion:

A feature of bash and tcsh (and possibly others) you can use the **TAB** key to complete a partially typed filename. For example if you have a file called constantine-monks-and-willy-wonka.txt in your directory and want to edit it you can type 'vi const', hit the **TAB** key, and the shell will fill in the rest of the name for you (provided the completion is unique).

Bash will even complete the name of commands and environment variables. And if there are multiple completions, if you hit TAB twice bash will show you all the completions. Bash is the default user shell for most Linux systems.

## Redirection:

<b>grep string filename &gt; newfile</b>	Redirects the output of the above grep command to a file 'newfile'.
------------------------------------------	---------------------------------------------------------------------

`grep string filename >> existfile` Appends the output of the grep command to the end of 'existfile'.

The redirection directives, > and >> can be used on the output of most commands to direct their output to a file.

## Pipes:

The pipe symbol "|" is used to direct the output of one command to the input of another.

**For example:**

`ls -l | more`

This command takes the output of the long format directory list command "ls -l" and pipes it through the more command (also known as a filter). In this case a very long list of files can be viewed a page at a time.

`du -sc * | sort -n | tail`

The command "du -sc" lists the sizes of all files and directories in the current working directory. That is piped through "sort -n" which orders the output from smallest to largest size. Finally, that output is piped through "tail" which displays only the last few (which just happen to be the largest) results.

## Command Substitution

You can use the output of one command as an input to another command in another way called command substitution. Command substitution is invoked when enclosing the substituted command in backwards single quotes.

For example:

`cat `find . -name aaa.txt``

which will cat ( dump to the screen ) all the files named aaa.txt that exist in the current directory or in any subdirectory tree.

## Searching for strings in files: The grep command

`grep string filename` prints all the lines in a file that contain the string

## Searching for files : The find command

`find search_path -name filename`

<code>find . -name aaa.txt</code>	Finds all the files named <code>aaa.txt</code> in the current directory or any subdirectory tree.
<code>find / -name vimrc</code>	Find all the files named <code>'vimrc'</code> anywhere on the system.
<code>find /usr/local/games -name "*xpilot*"</code>	Find all files whose names contain the string <code>'xpilot'</code> which exist within the <code>'/usr/local/games'</code> directory tree.

## Reading and writing backups and archives: The tar command

The tar command is the "standard" way to read and write archives (collections of files and whole directory trees).

Often you will find archives of stuff with names like `stuff.tar`, or `stuff.tar.gz`. This is stuff in a tar archive, and stuff in a tar archive which has been compressed using the gzip compression program respectively.

Chances are that if someone gives you a tape written on a LINUX system, it will be in tar format, and you will use tar (and your tape drive) to read it.

Likewise, if you want to write a tape to give to someone else, you should probably use tar as well.

Tar examples:

<code>tar xv</code>	extracts (x) files from the default tape drive while listing (v = verbose) the file names to the screen.
<code>tar tv</code>	lists the files from the default tape device without extracting them.
<code>tar cv file1 file2</code>	Write files <code>'file1'</code> and <code>'file2'</code> to the default tape device.
<code>tar cvf archive.tar file1 [file2...]</code>	create a tar archive as a file <code>"archive.tar"</code> containing <code>file1</code> , <code>file2...etc.</code>
<code>tar xvf archive.tar</code>	extract from the archive file
<code>tar cvfz archive.tar.gz dname</code>	create a gzip compressed tar archive containing everything in the directory <code>'dname'</code> . This does not work with all versions of tar.
<code>tar xvfz archive.tar.gz</code>	extract a gzip compressed tar archive. Does not work with all versions of tar.
<code>tar cvfI archive.tar.bz2 dname</code>	Create a bz2 compressed tar archive. Does not work with all versions of tar

## File compression: compress, gzip, and bzip2

A common compression utility is gzip (and gunzip). These are the GNU compress and uncompress utilities. gzip usually gives better compression than standard compress, but may not be installed on all systems. The suffix for gzipped files is .gz

**gzip part** Creates a compressed file part.gz **gunzip part.gz** Extracts the original file from part.gz

The bzip2 utility has (in general) even better compression than gzip, but at the cost of longer times to compress and uncompress the files. It is not as common a utility as gzip, but is becoming more generally available.

**bzip2 part** Create a compressed bzip2 file par.bz2 **bunzip2 part.bz2** Uncompress the compressed part file.

## Looking for help: The man commands

Most of the commands have a manual page which give sometimes useful, often more or less detailed, sometimes cryptic and unfathomable descriptions of their usage. Some say they are called man pages because they are only for real men.

Example:

**man ls** Shows the manual page for the ls command

## Basics of the vi editor

**vi filename** Opening a file

Creating text

Edit modes: These keys enter editing modes and type in the text of your document.

**i** Insert before current cursor position **I**  
Insert at beginning of current line **a**  
Insert (append) after current cursor position  
**A** Append to end of line **x**  
Replace 1 character

**R** Replace mode  
**<ESC>** Terminate insertion or overwrite mode

#### Deletion of text

**x** Delete single character  
**dd** Delete current line and put in buffer  
**ndd** Delete n lines (n is a number) and put them in buffer  
**J** Attaches the next line to the end of the current line (deletes carriage return).  
**u** Undo last command

#### Cut and paste

**yy** Yank current line into buffer **nyy** Yank n lines into buffer  
**p** Put the contents of the buffer after the current line **P**  
Put the contents of the buffer before the current line

#### Cursor positioning

**^d** Page down  
**^u** Page up  
**:n** Position cursor at line n  
**:\$** Position cursor at end of file  
**^g** Display current line number

**h, j, k, l** **Left, Down, Up** and **Right** respectively. Your arrow keys should also work.

#### String substitution

**:n1,n2:s/string1/string2/[g]** Substitute string2 for string1 on lines n1 to n2. If g is included (meaning global), all instances of string1 on each line are substituted. If g is not included, only the first instance per matching line is substituted.

**^** matches start of line  
**.** matches any single character  
**\$** matches end of line

These and other "special characters" (like the forward slash) can be "escaped" with \ i.e to match the string "/usr/STRIM100/SOFT" say "\usr\STRIM100\SOFT"

#### Examples:

**:1,\$:s/dog/cat/g** Substitute 'cat' for 'dog', every instance for the entire file - lines 1 to \$ (end of file)

**:23,25:/frog/bird/** Substitute 'bird' for 'frog' on lines 23 through 25. Only the first instance on each line is substituted.



## Saving and quitting and other "ex" commands

These commands are all prefixed by pressing colon (:) and then entered in the lower left corner of the window. They are called "ex" commands because they are commands of the ex text editor - the precursor line editor to the screen editor vi. You cannot enter an "ex" command when you are in edit mode (typing text onto the screen) Press <ESC> to exit from an editing mode.

<b>:w</b>	Write the current file.
<b>:w new.file</b>	Write the file to the name 'new.file'.
<b>:w! existing.file</b>	Overwrite an existing file with the file currently being edited.
<b>:wq</b>	Write the file and quit.
<b>:q</b>	Quit.
<b>:q!</b>	Quit with no changes.
<b>:e filename</b>	Open the file 'filename' for editing.
<b>:set number</b>	Turns on line numbering
<b>:set nonumber</b>	Turns off line numbering