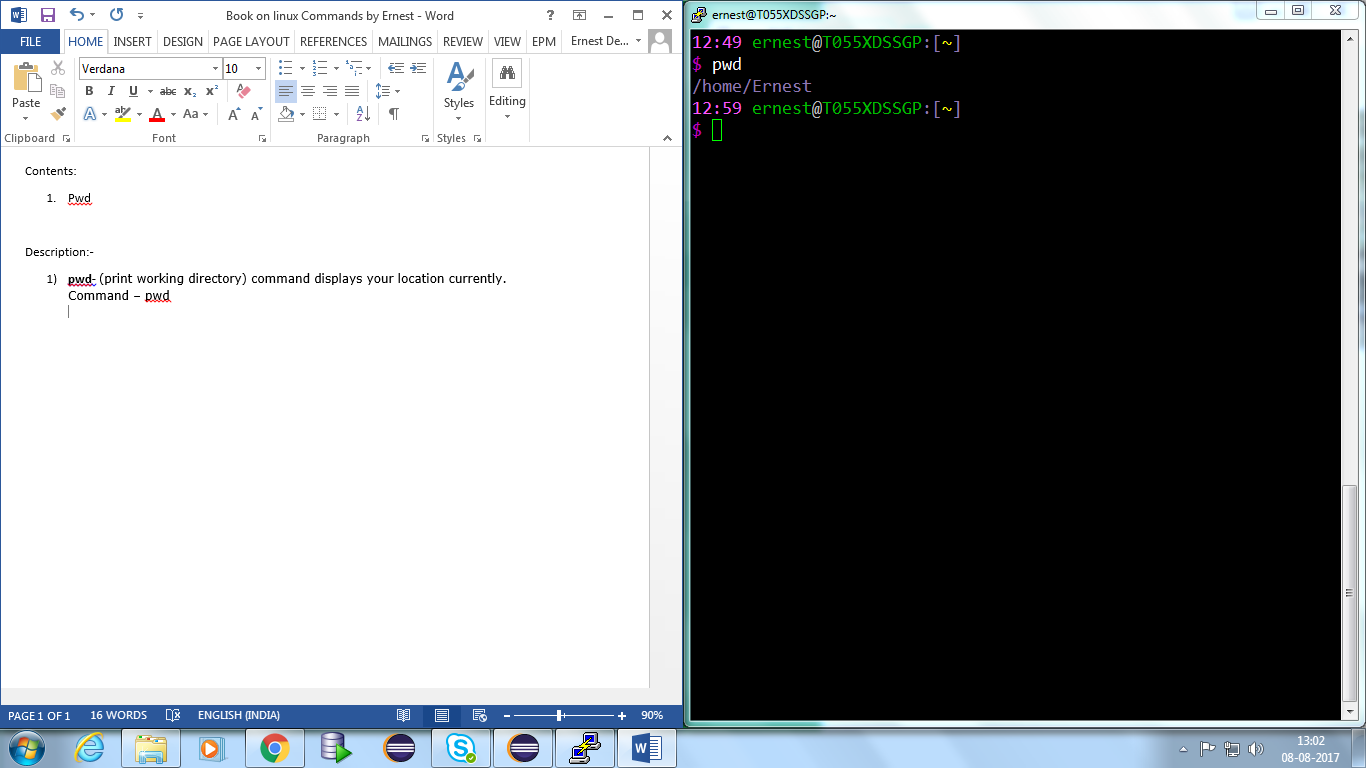
**Date:-08/08/2017**

**Basic Syntax**

**Description:-**

1. **pwd**- (print working directory) Syntax displays your location currently.

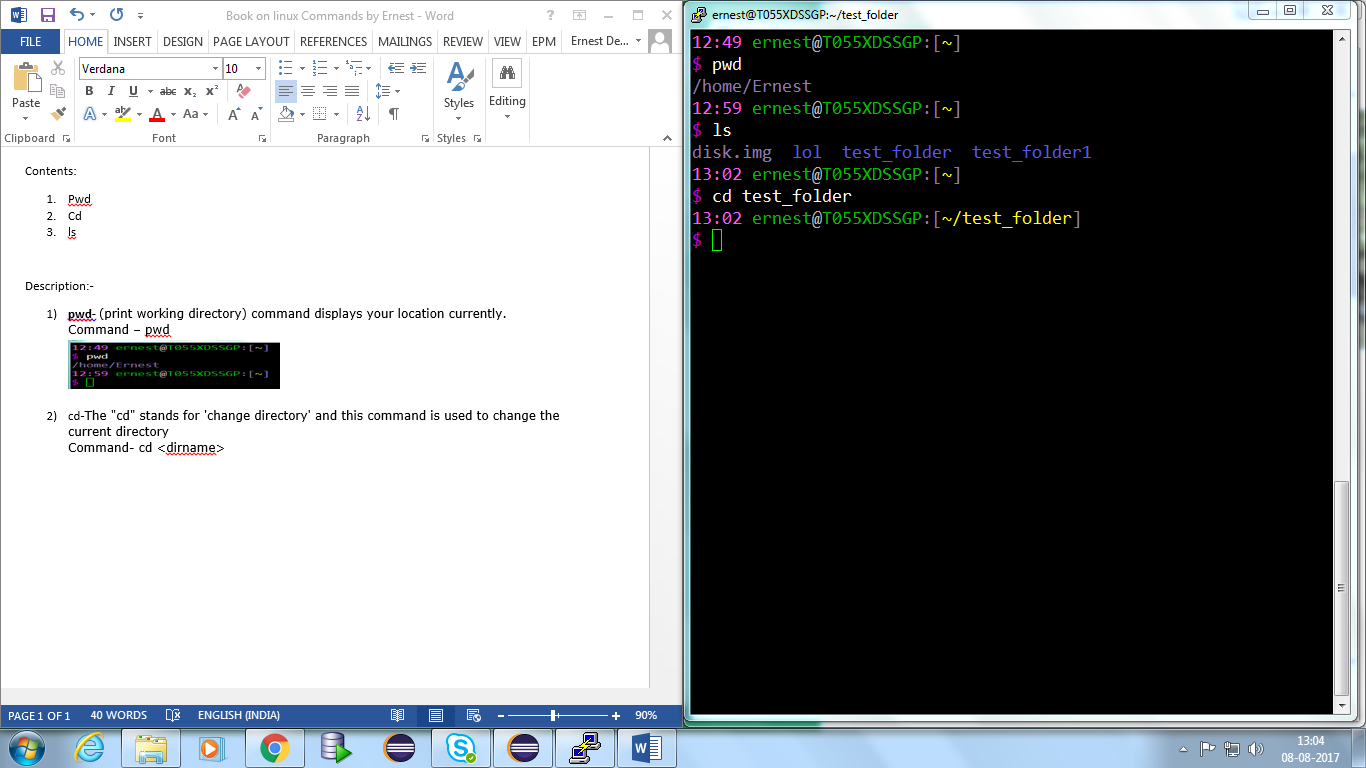
Syntax – pwd



1. **cd**-The "cd" stands for 'change directory' and this Syntax is used to change the current directory

Syntax- cd <dirname>

Example:- cd /home/Ernest/test\_folder



1. **ls**- It will show the full list or content of your directory.

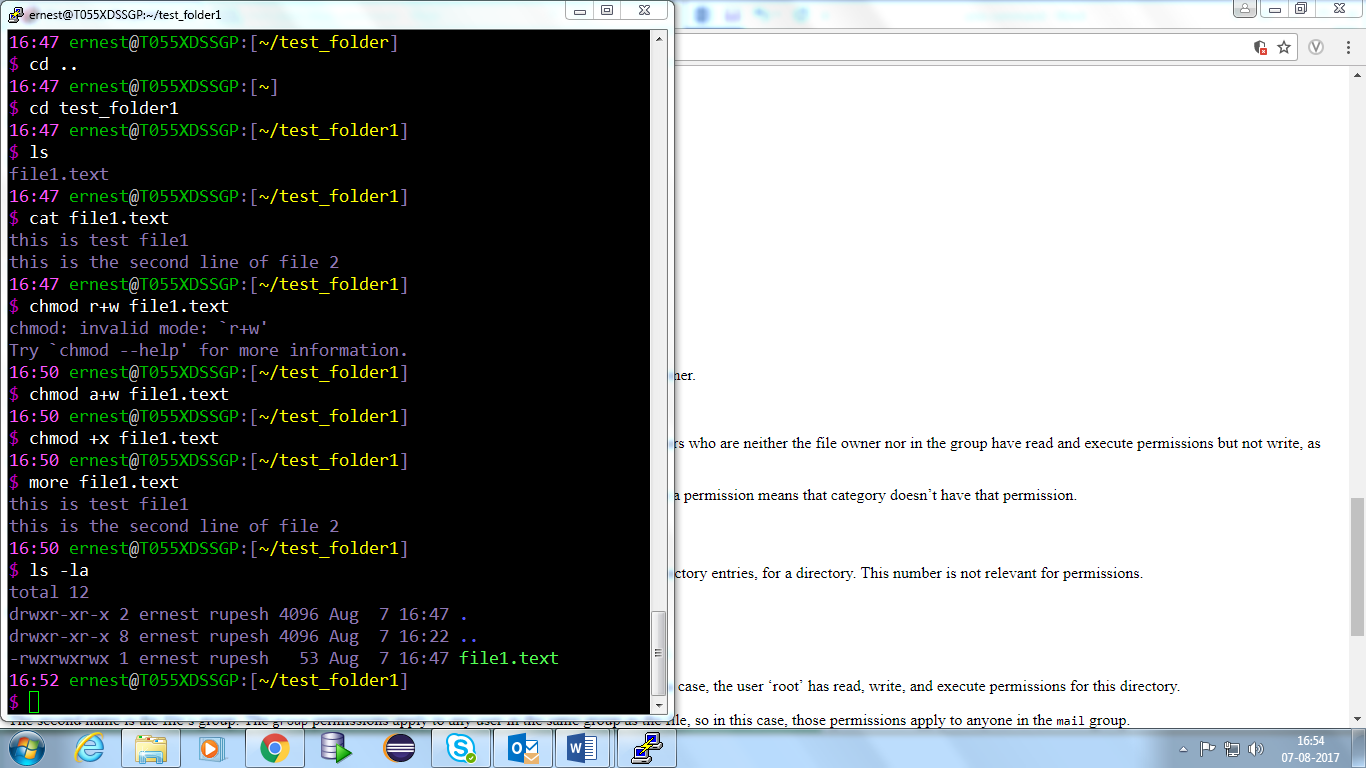
Syntax :- ls <options>

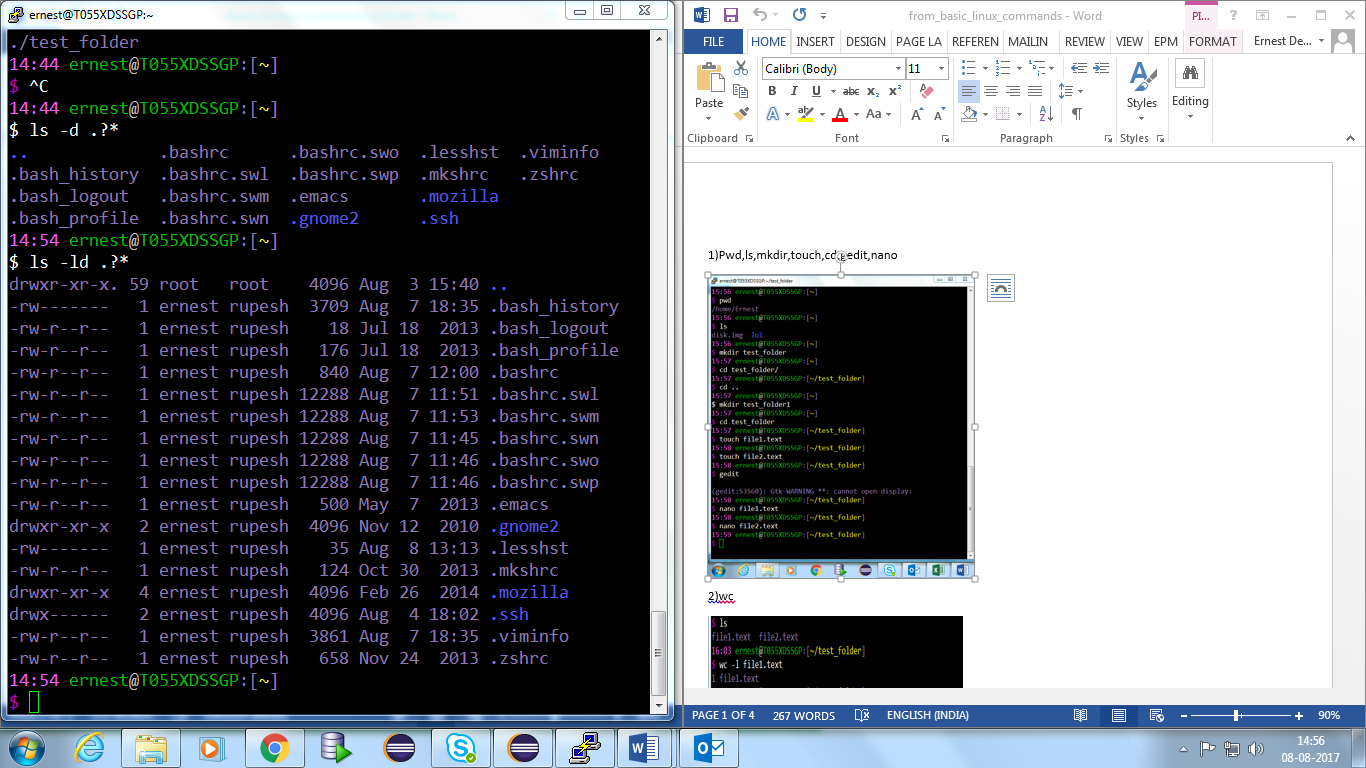
ls –d .?\* - displays hidden files

ls –ld .?\* - displays hidden files along with other details like permissions.

ls –a - list all files

ls –lrth





1. **Find**-To search a file

**Syntax**- find name foldername

**Example-**

find -iname test\_folder

Find . -mindepth 1 -maxdepth 1 -name '.\*'

Or

find . -mindepth 1 -maxdepth 1 -name '.\*' -o -name '\*~'

find . searches current directory

-mindepth 1 excludes . and .. from the list

-maxdepth 1 limits the search to the current directory

-name '.\*' find file names that start with a dot

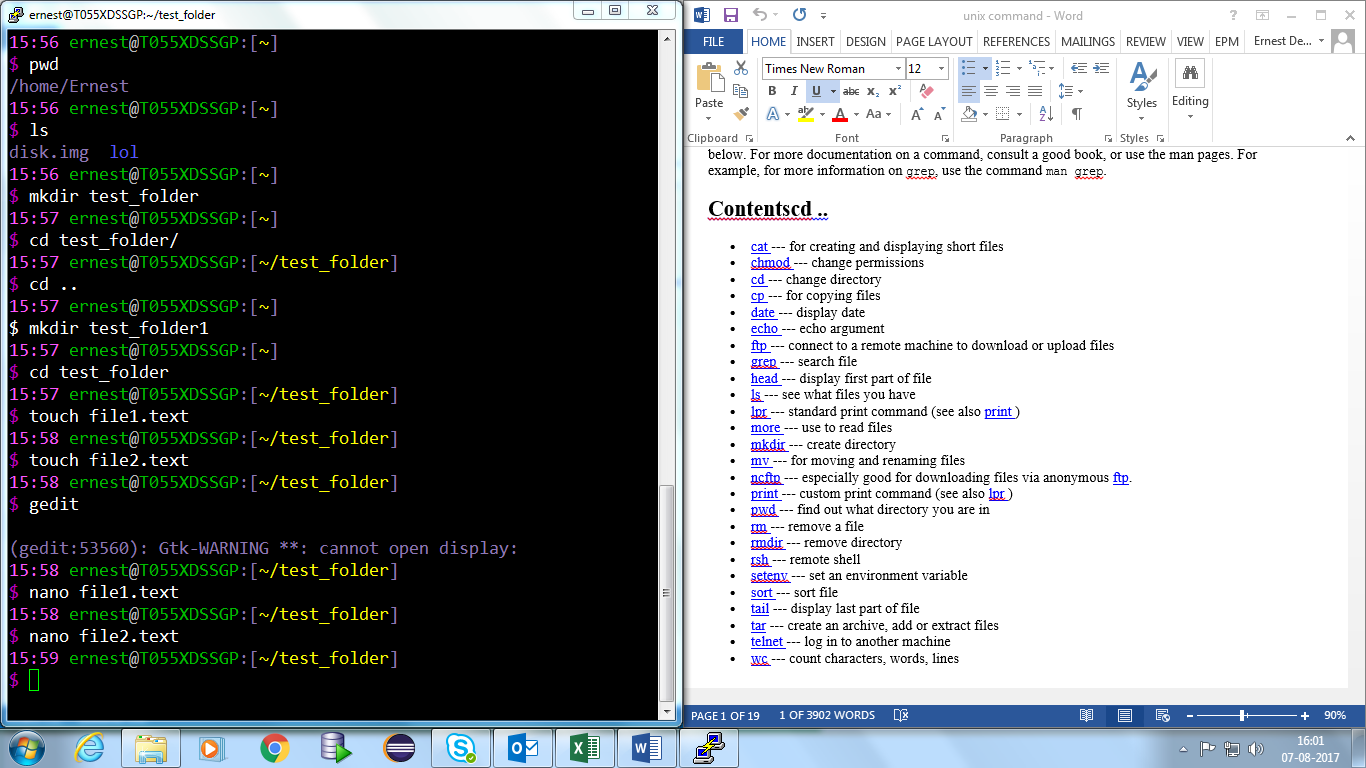
-o or

-name '\*~' find file names that end with a tilde (usually, these are backup files from text editing programs)



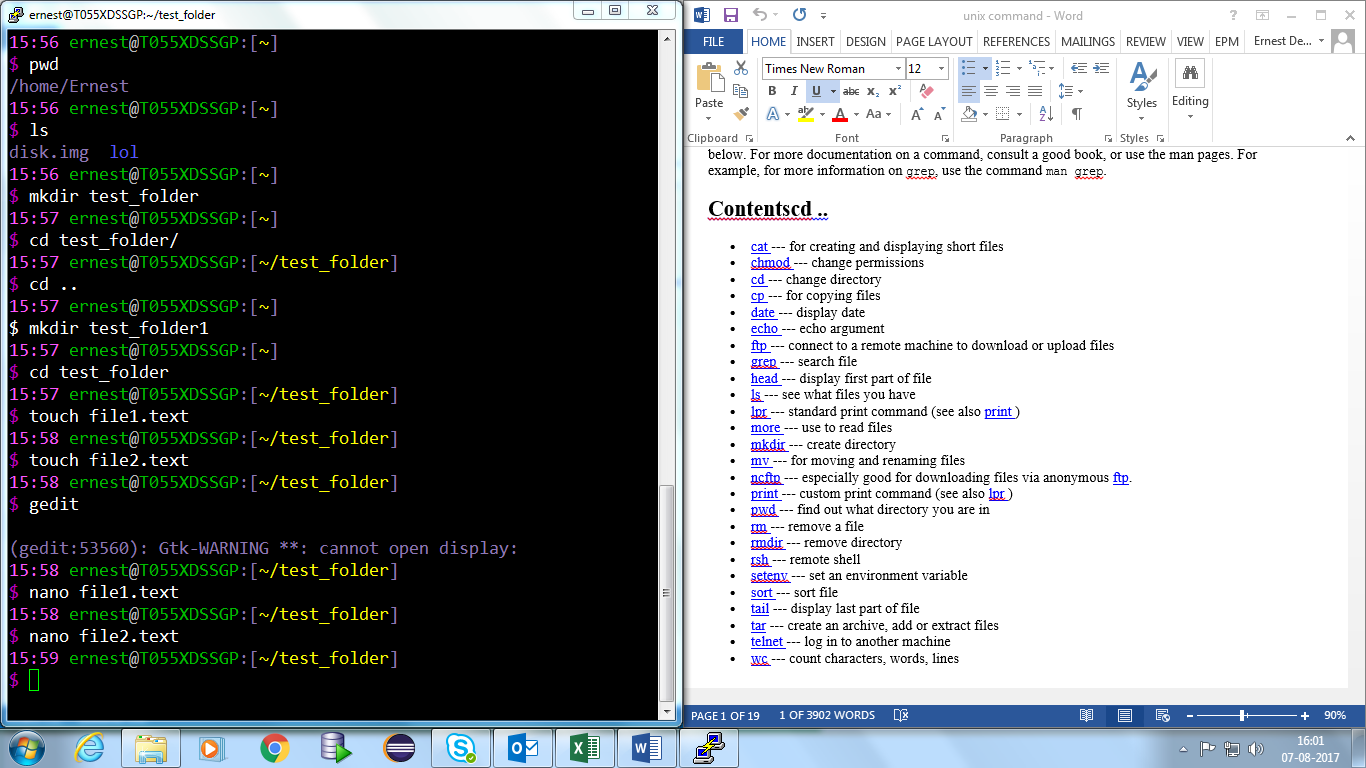
1. **Mkdir** – to make a directory/folder.

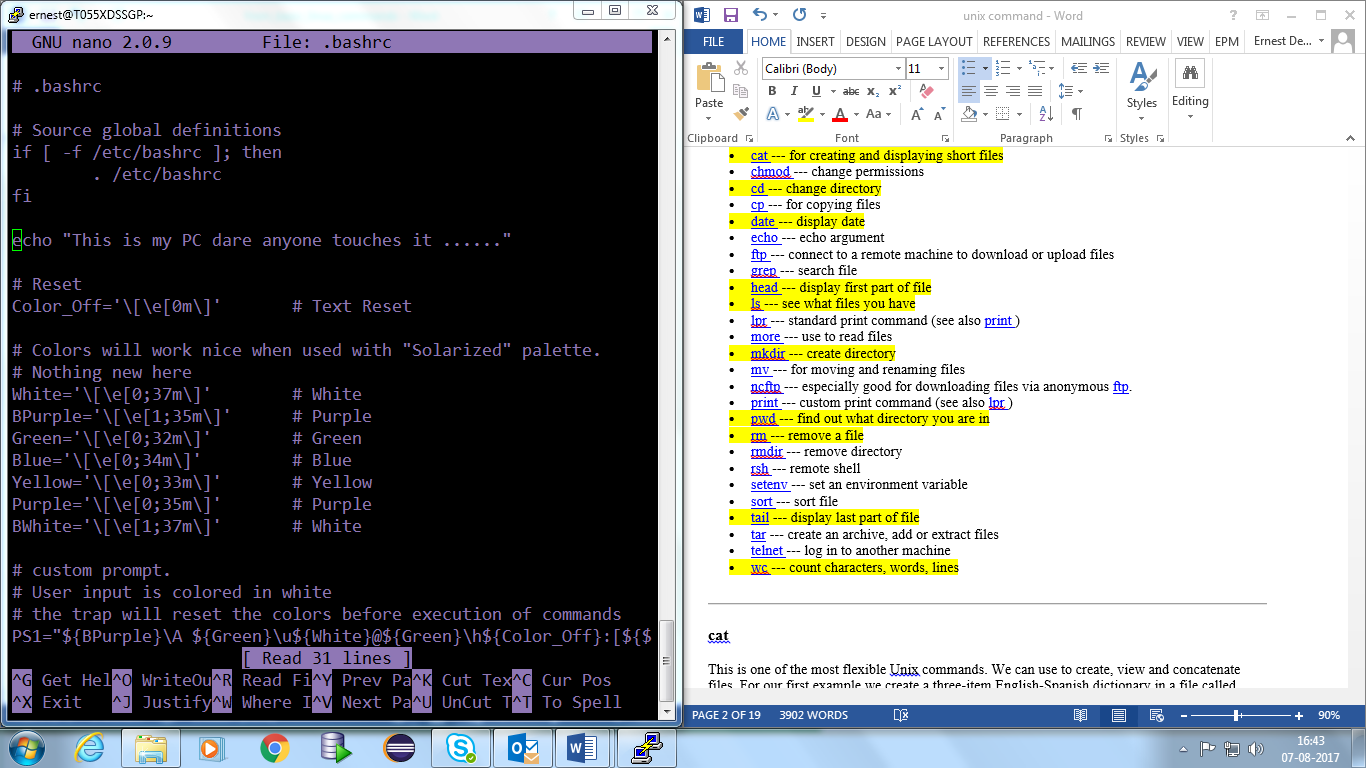
Syntax-mkdir



1. **Nano**-default editor

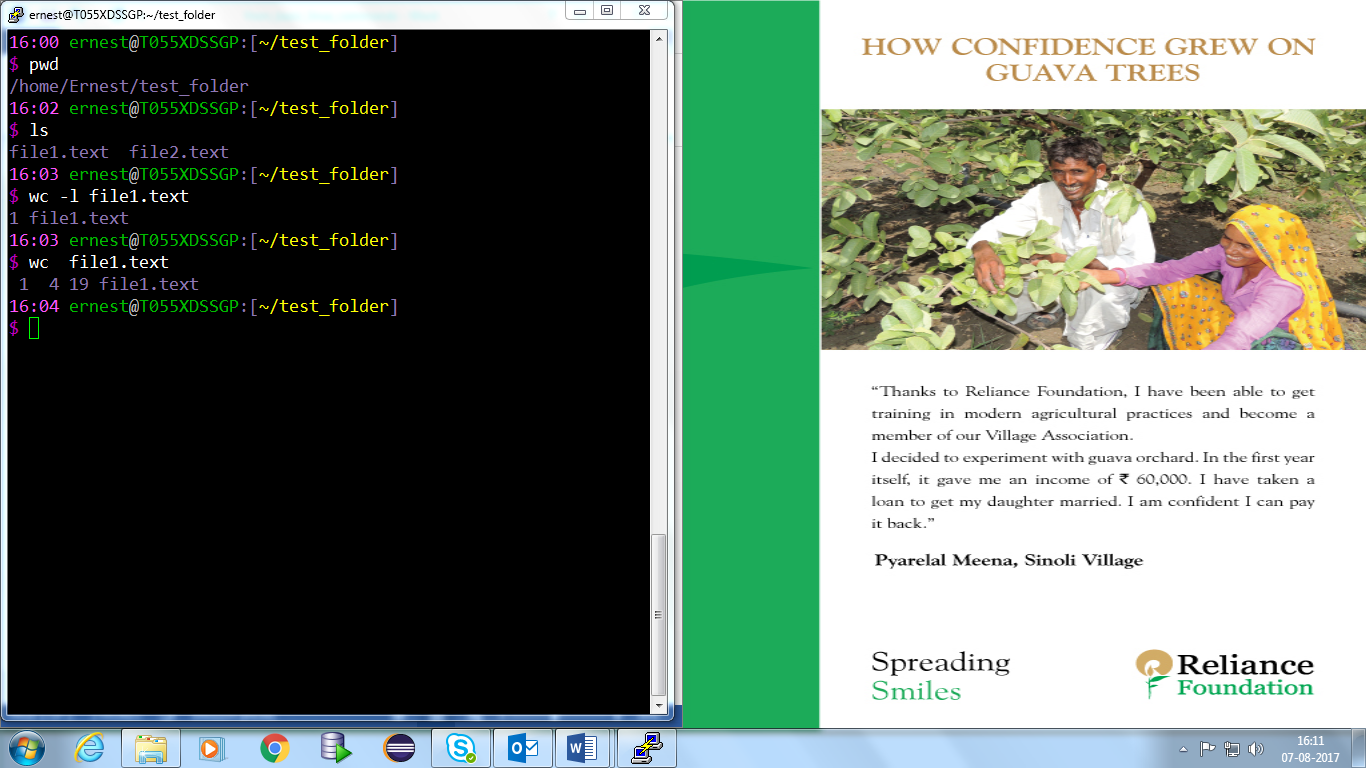
Syntax:- nano <file\_name>





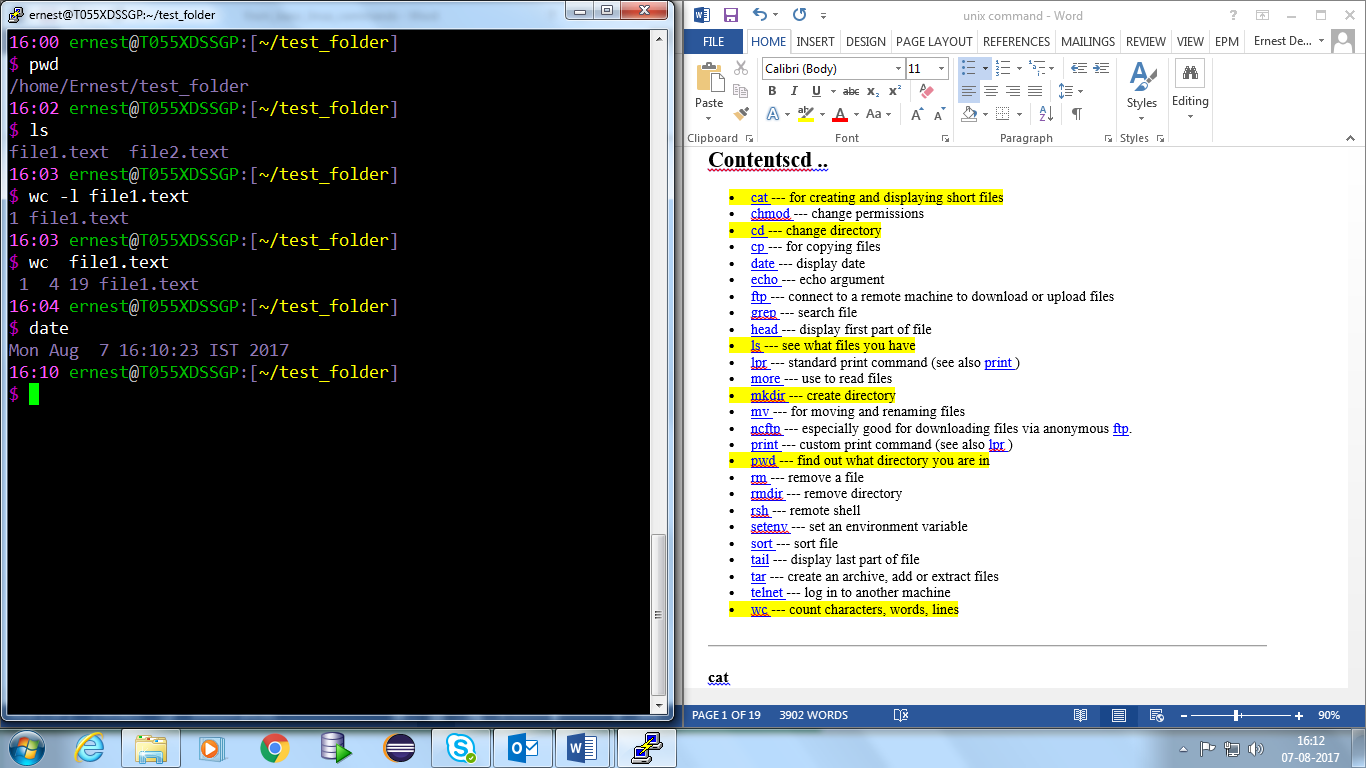
1. **Wc**-Word Count (To count the number of words).

**Syntax**- wc <options> <folder\_name>



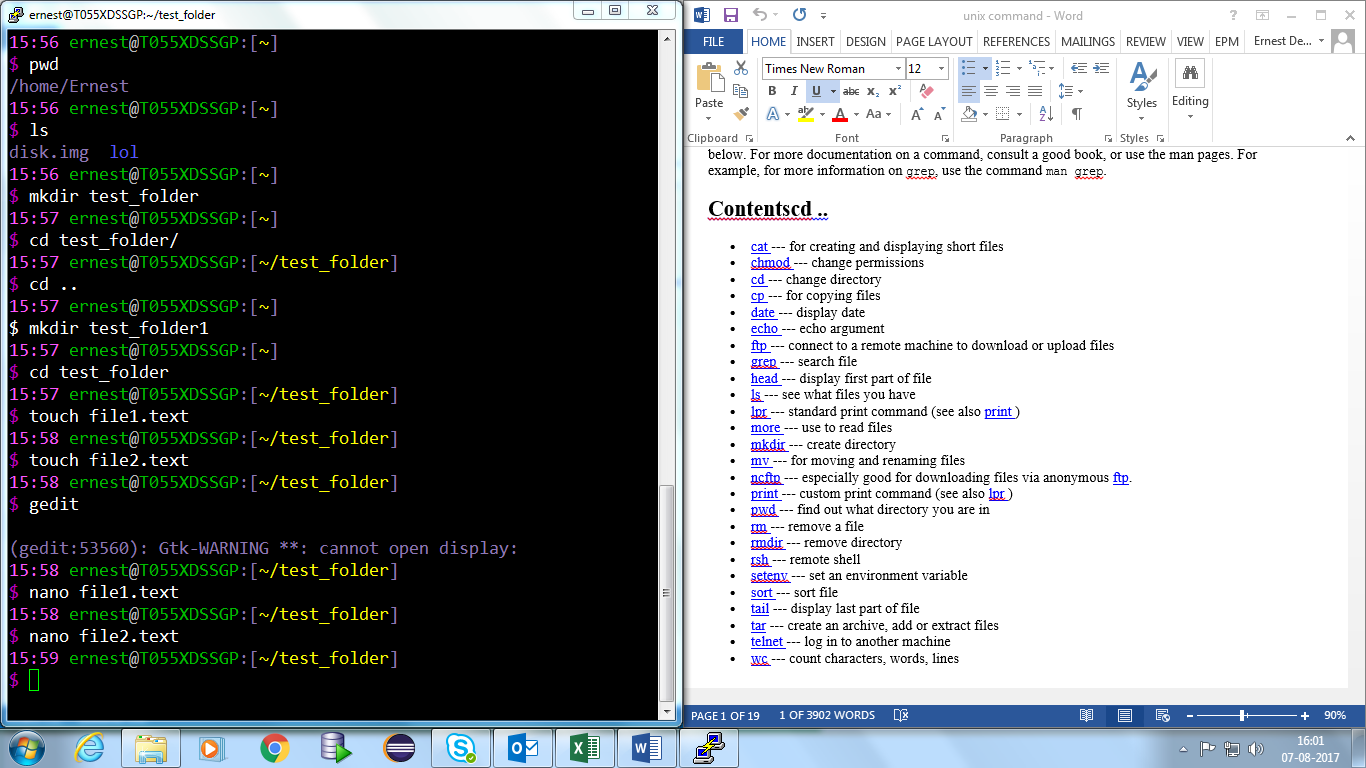
1. **Date**- to display the date

Syntax- date



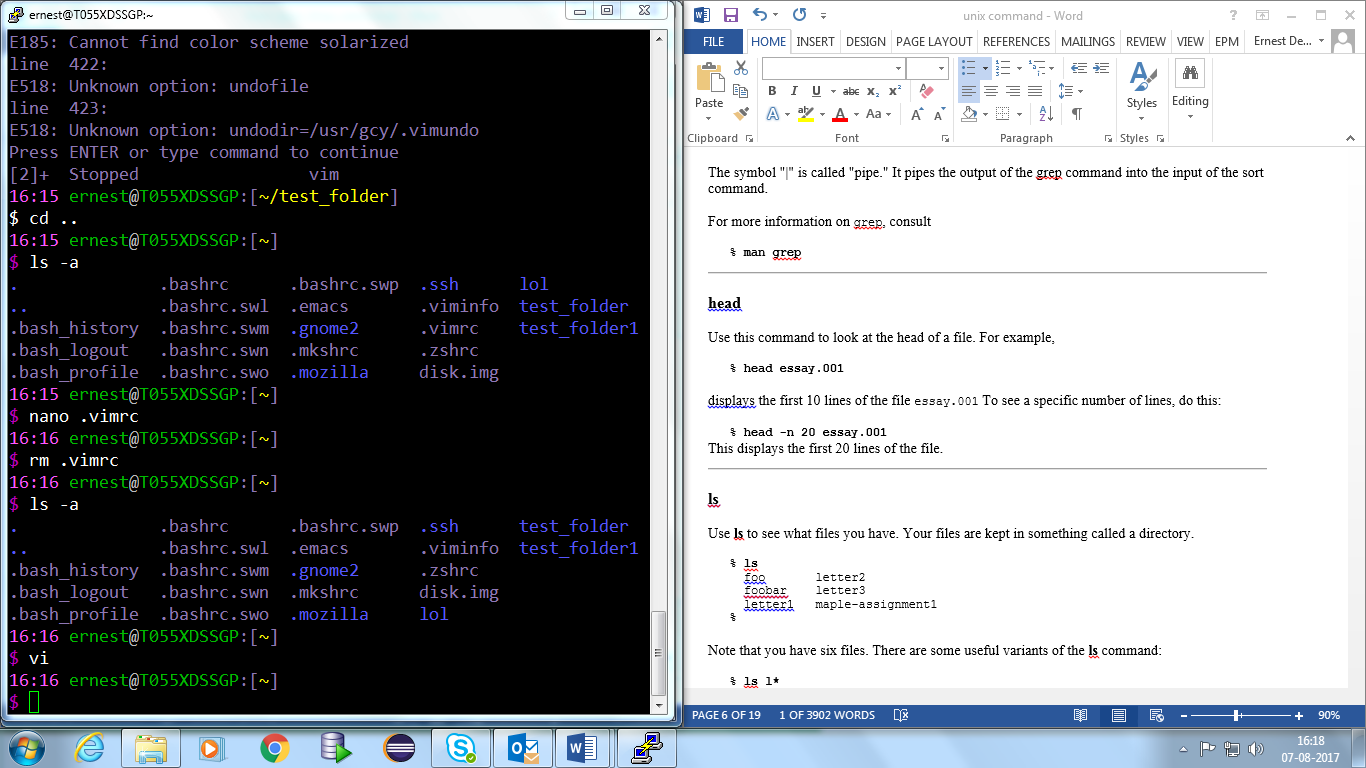
1. **Touch** – To create a file

Syntax- touch <file\_name>



1. **rm**- To remove a file.

**Syntax**:- rm <options> <filename>

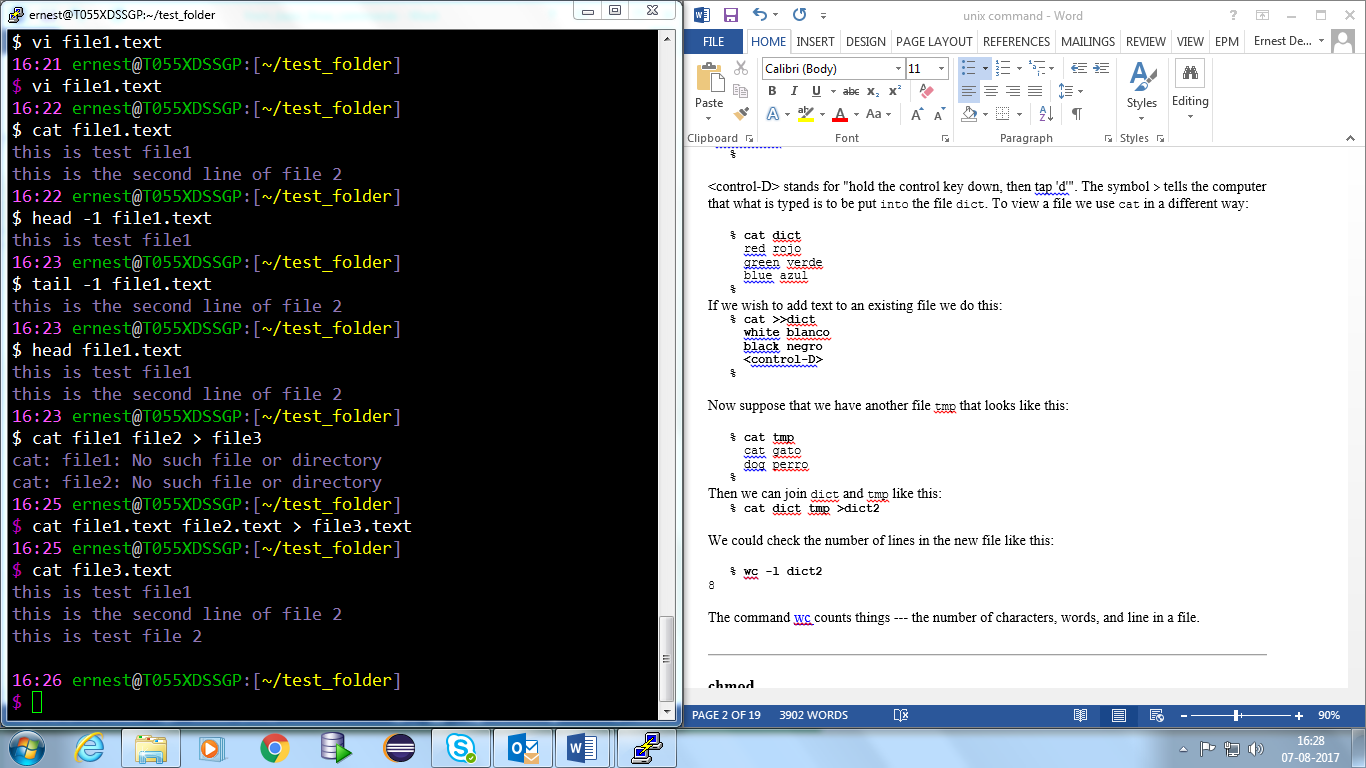


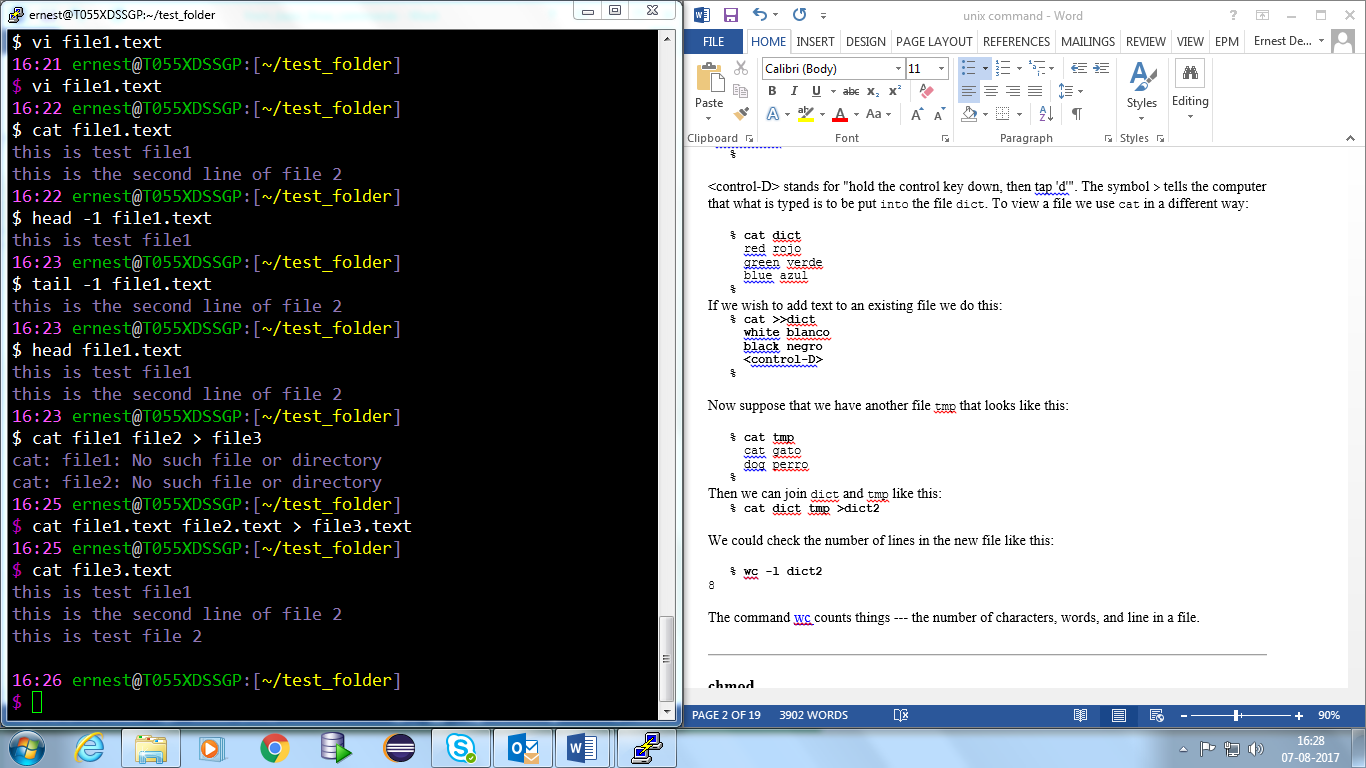
1. **Cat**-It can be used to concatenate to files or display the contents of the file.

**Syntax**- cat <file\_name>

**Example**:- cat file1.text

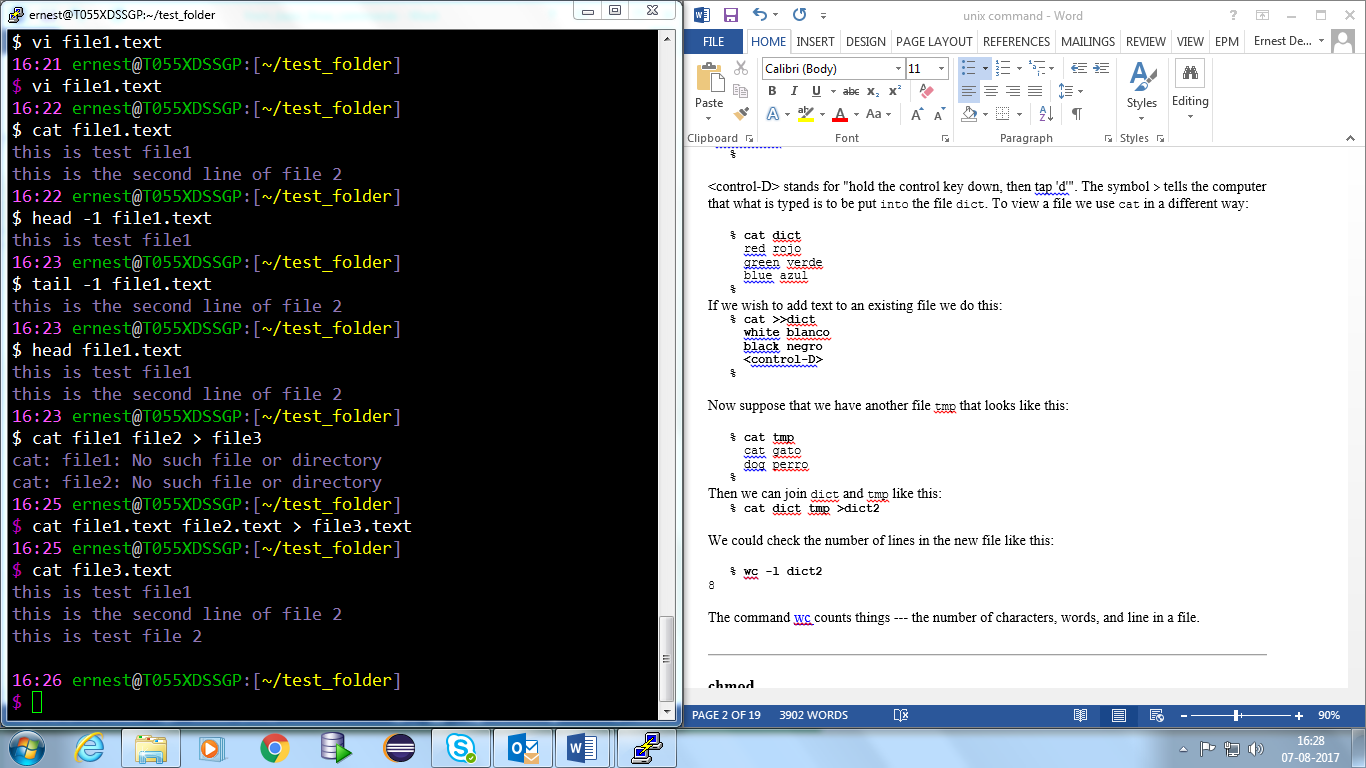
cat file1.text file2.text > file3.text (This concatenates the contents of file1,2 to file 3)





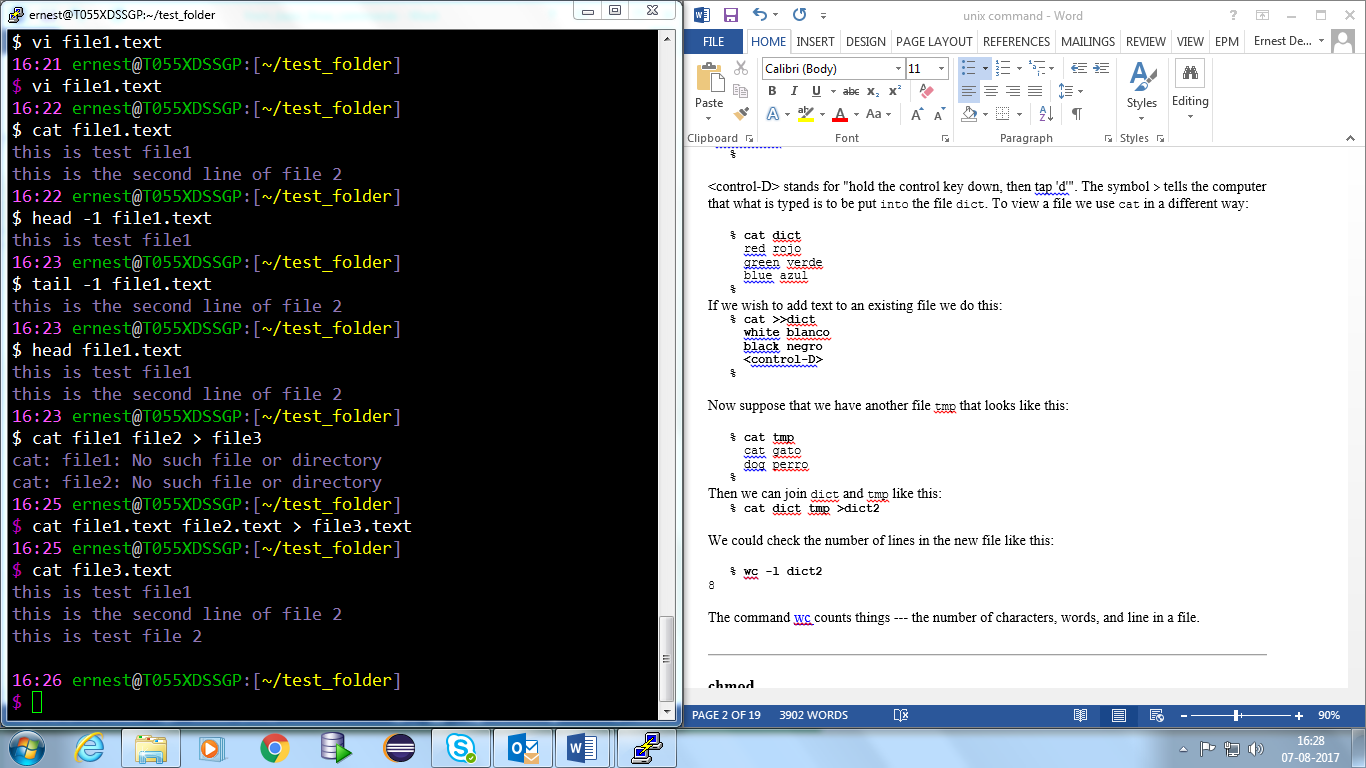
1. **head** –  Prints the first 10 lines of each FILE to standard output.

Syntax:- head <options> <file\_name>



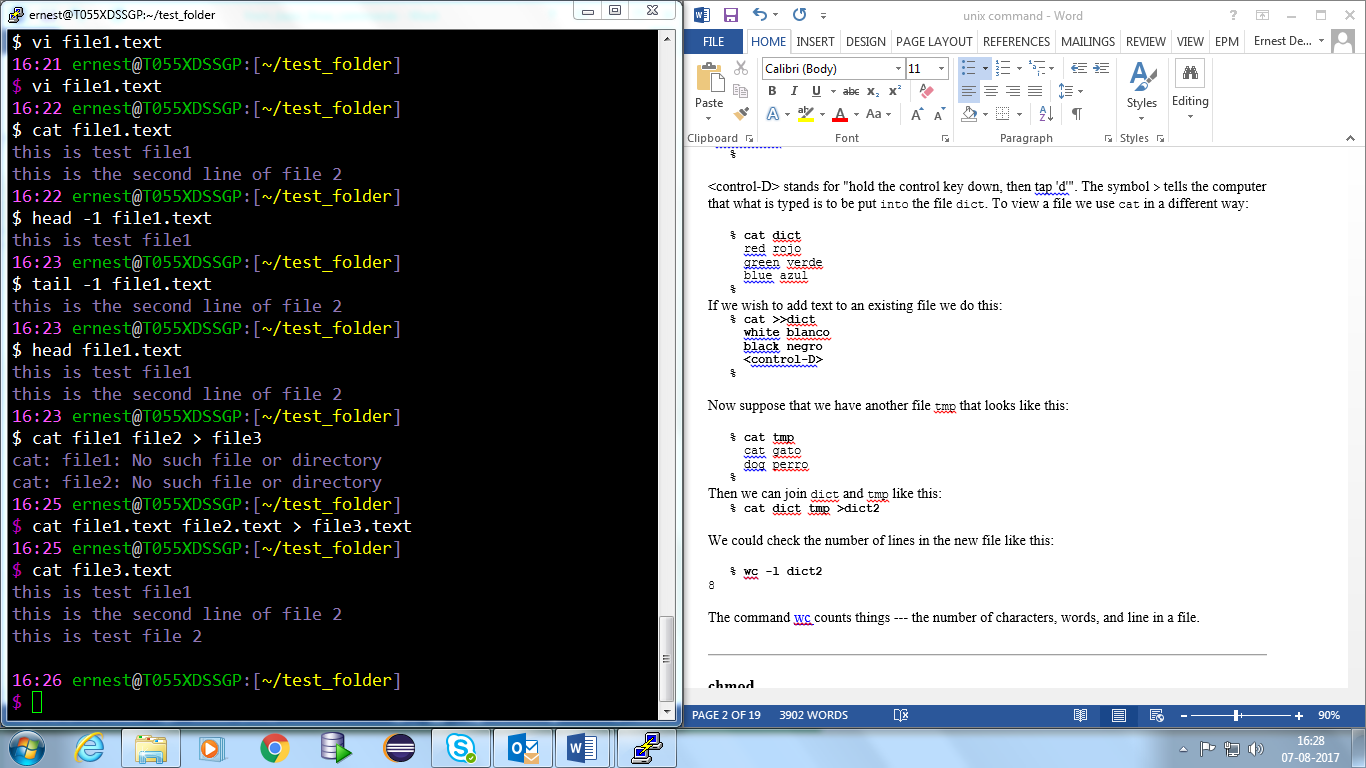
1. **tail**-By default “tail” prints the last 10 lines of a file, then exits

Syntax- tail <options> <file\_name>



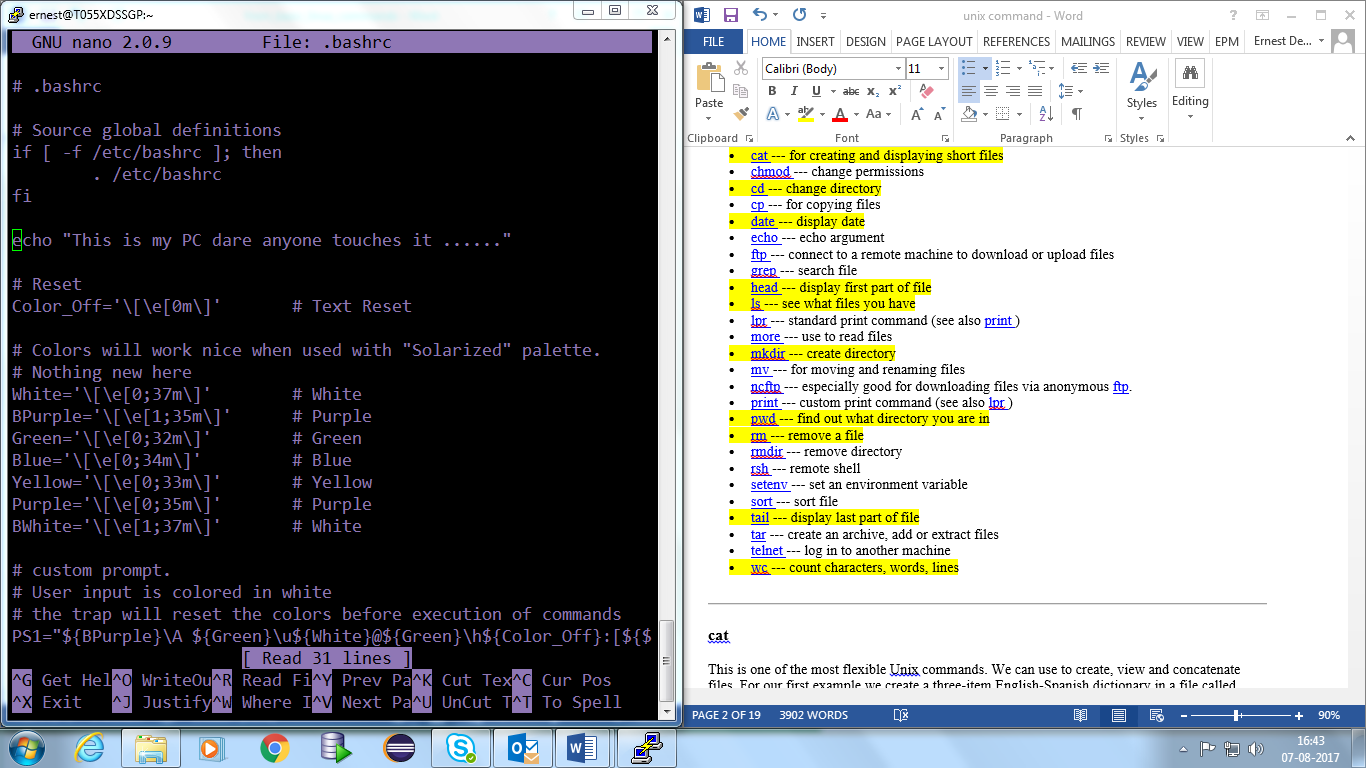
1. **vi**- Editor

Syntax – vi <file\_name>



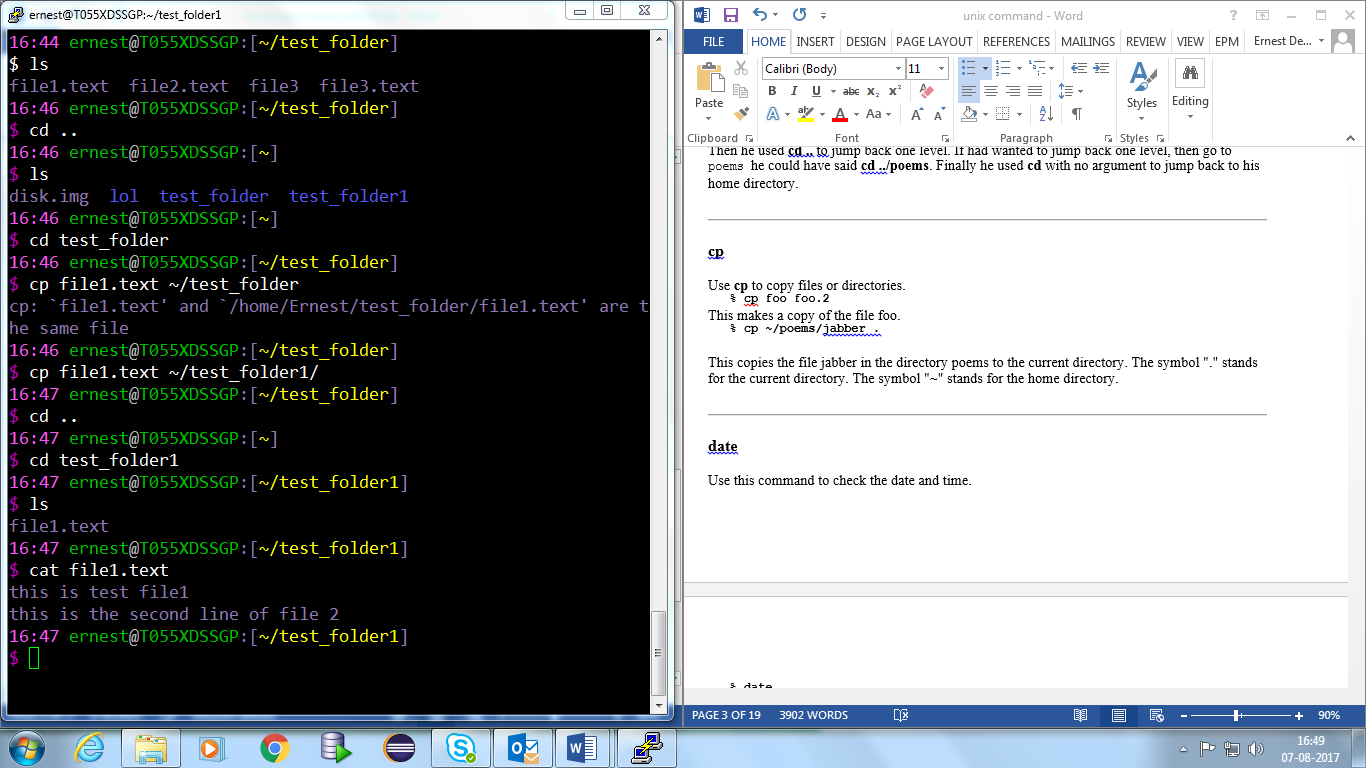
1. **Echo-** Used to print a variable or any text

Syntax: echo “<Enter Your text here>”



1. cp- To copy files.

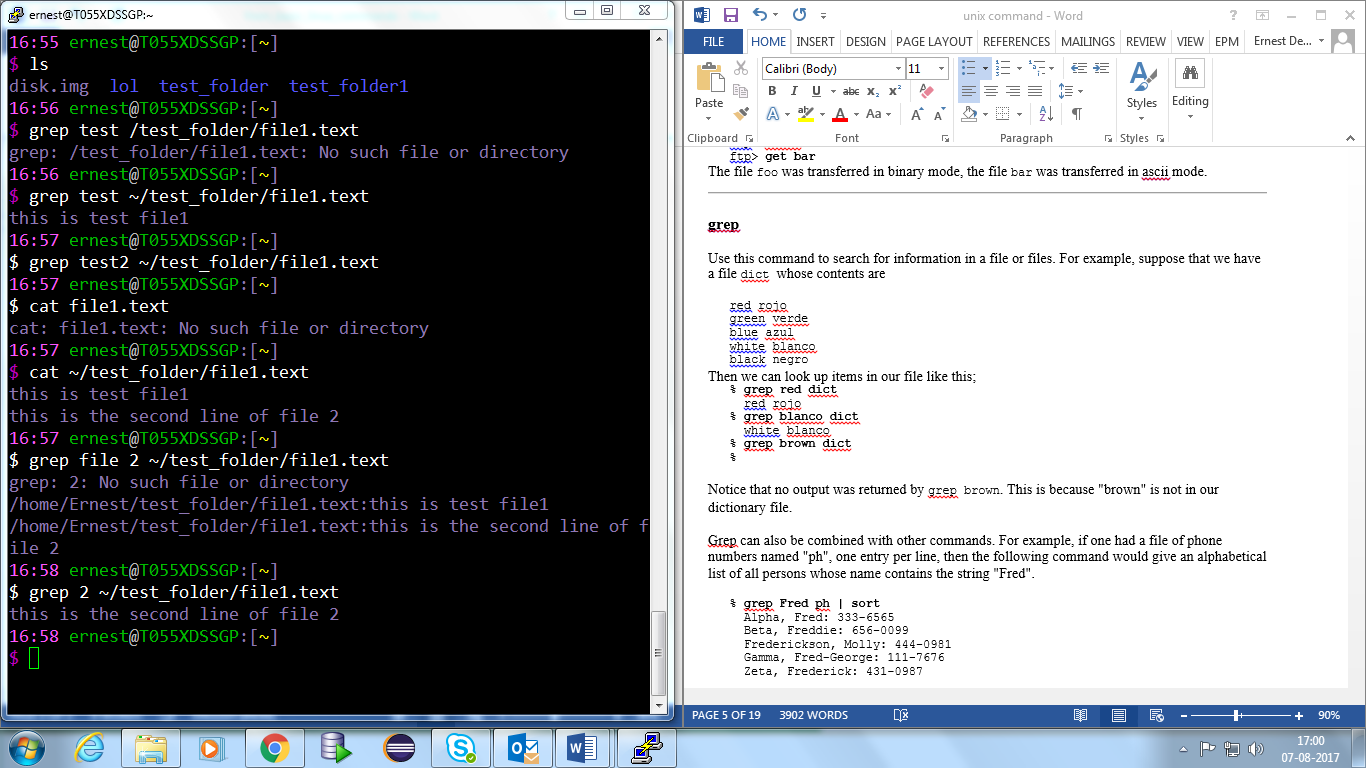
Syntax – cp <file\_name>



1. grep - searches for matching a regular expression against text in a file, multiple files or a stream of input. It searches for the pattern of text that you specify on the**Syntax** line and prints output for you.

Syntax- grep "literal\_string" filename.

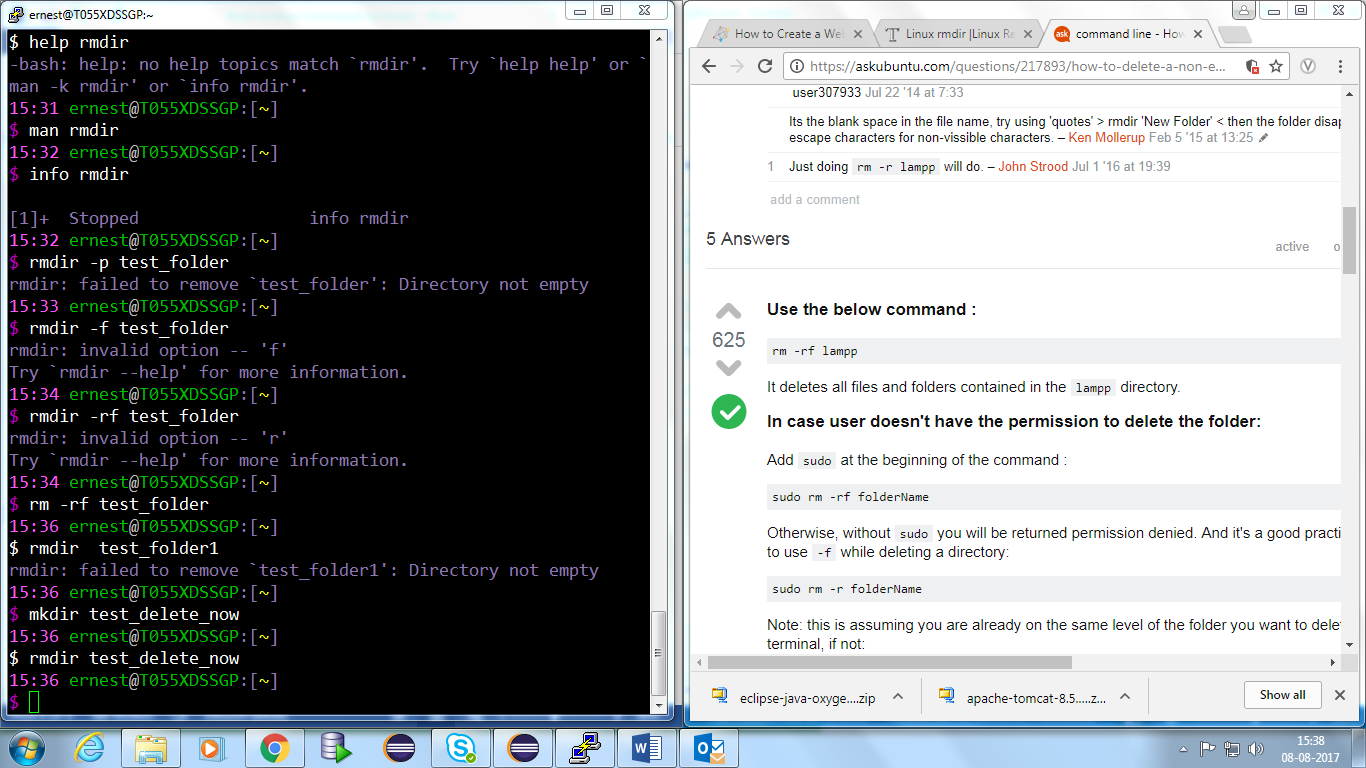
Example- grep "this" demo\_file



1. rmdir- To remove a directory.

Syntax:- rmdir <options><directory/folder\_name>

example:- rmdir –p test\_folder (deletes the subfolders too inside test\_folder.



Note- In case there are files inside the use

rm -rf <folder\_name>

**Monitoring Commands**

1. **Top**:- cpu,memory,swap usage

Syntax- top



1. **vmstat**- **vmstat** (Virtual Memory Statistics) outputs instantaneous reports about your system's processes, memory, paging, block I/O, interrupts and CPU activity.

Syntax- vmstat <option>



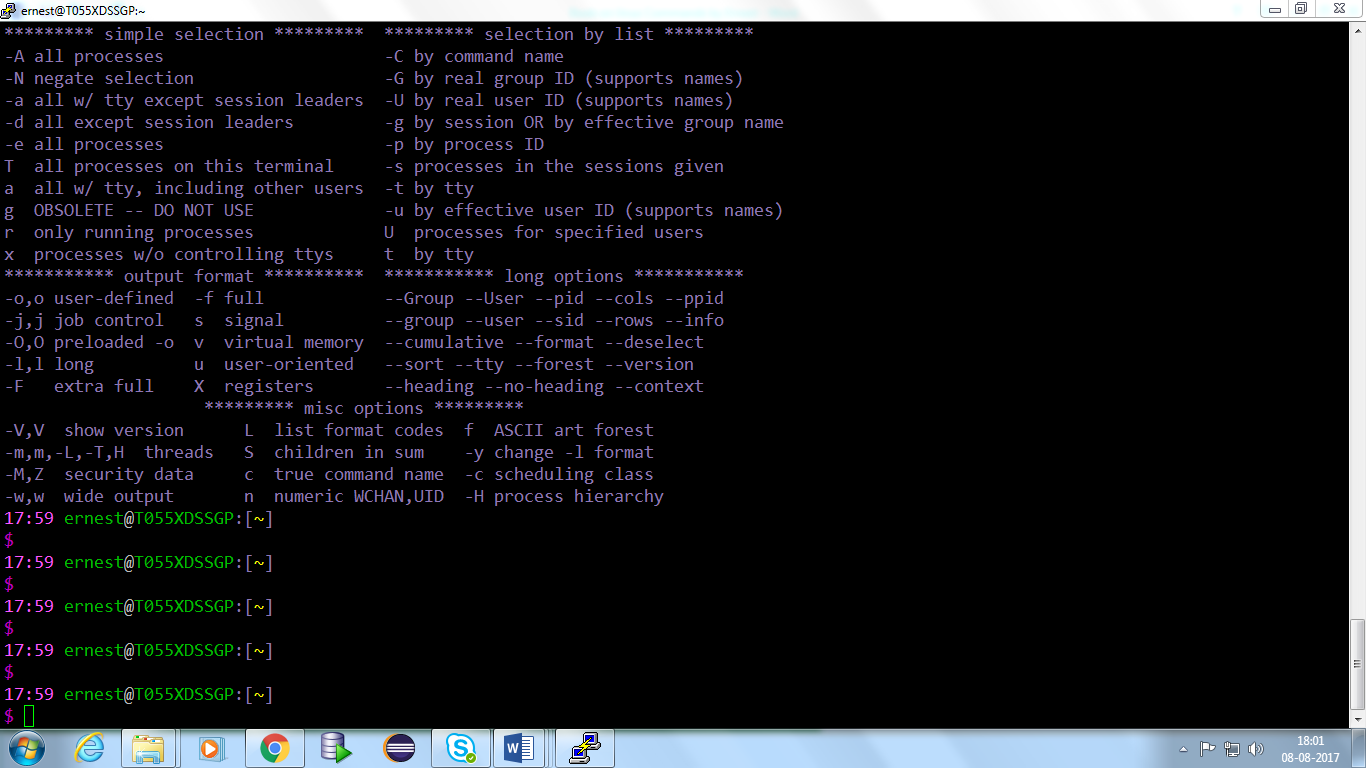
1. **ps** – Active processes

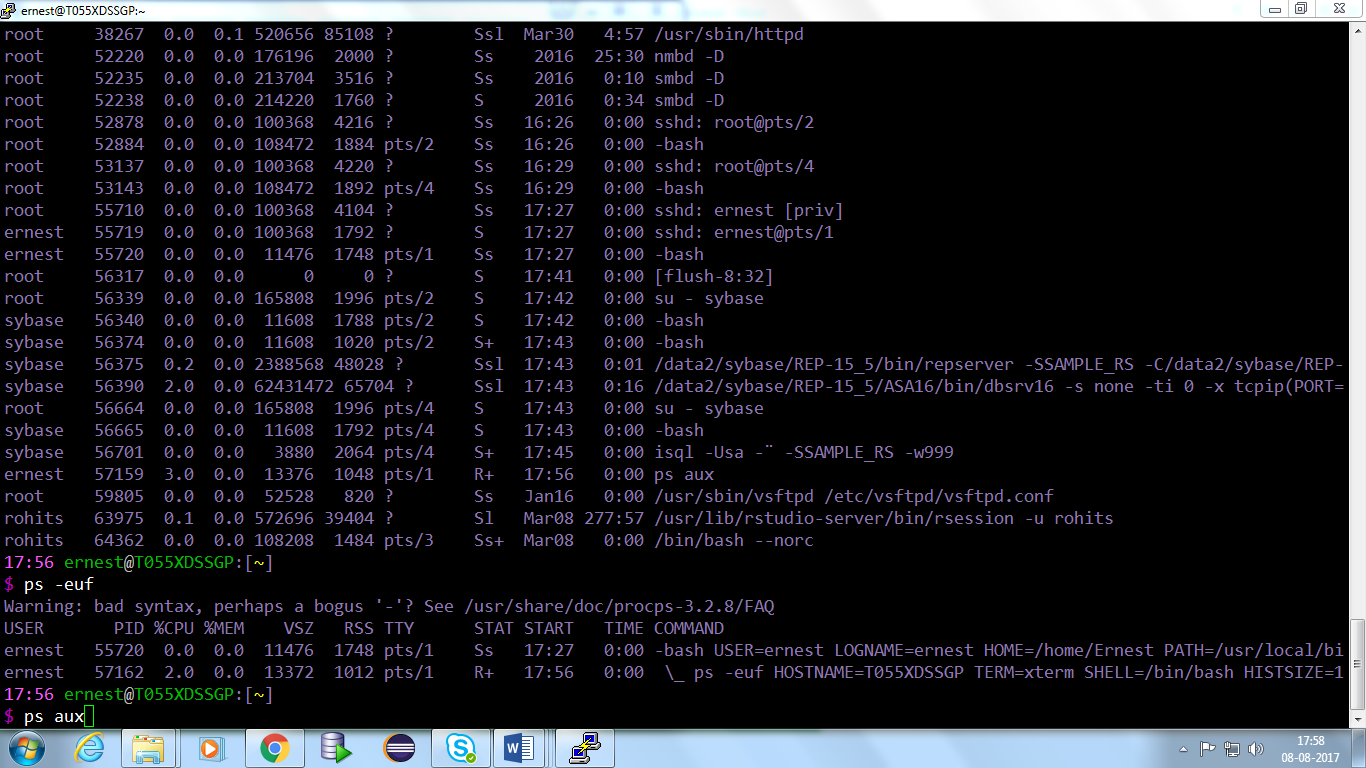
Syntax- ps <option>

Example-

$ ps aux

$ ps –euf



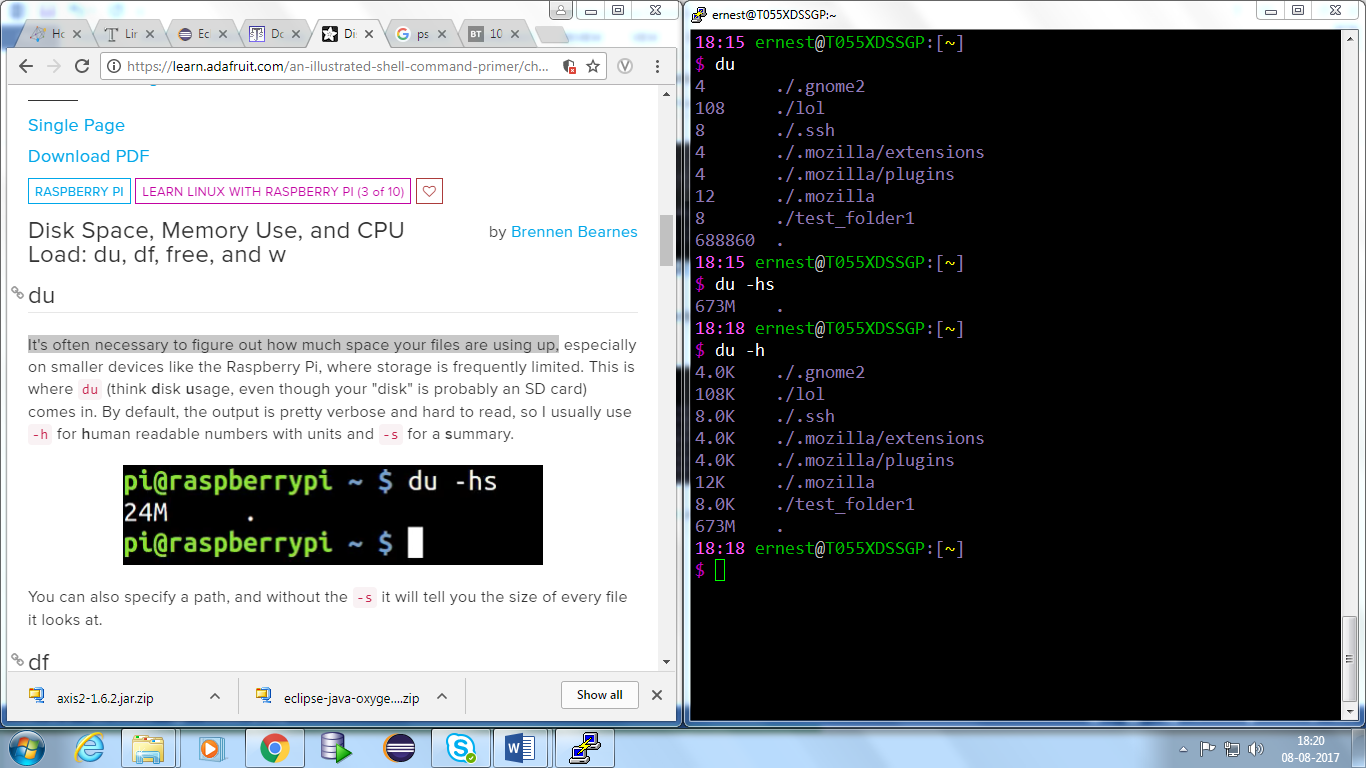


1. **sar** = (System Activity Reporter) collects and reports information about today's system activity so far.

**Syntax:- sar <options>**

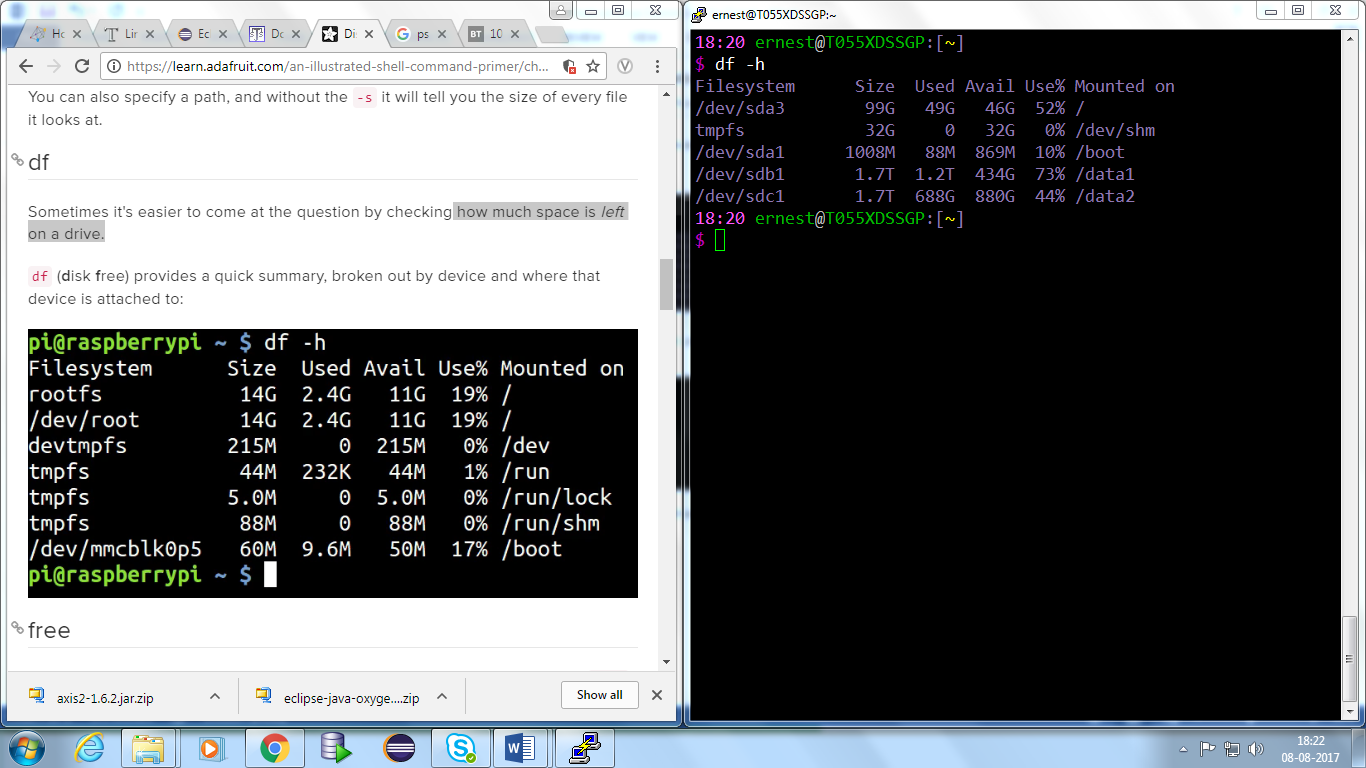
1. **du**- how much space your files are using up.

Syntax- du <options>



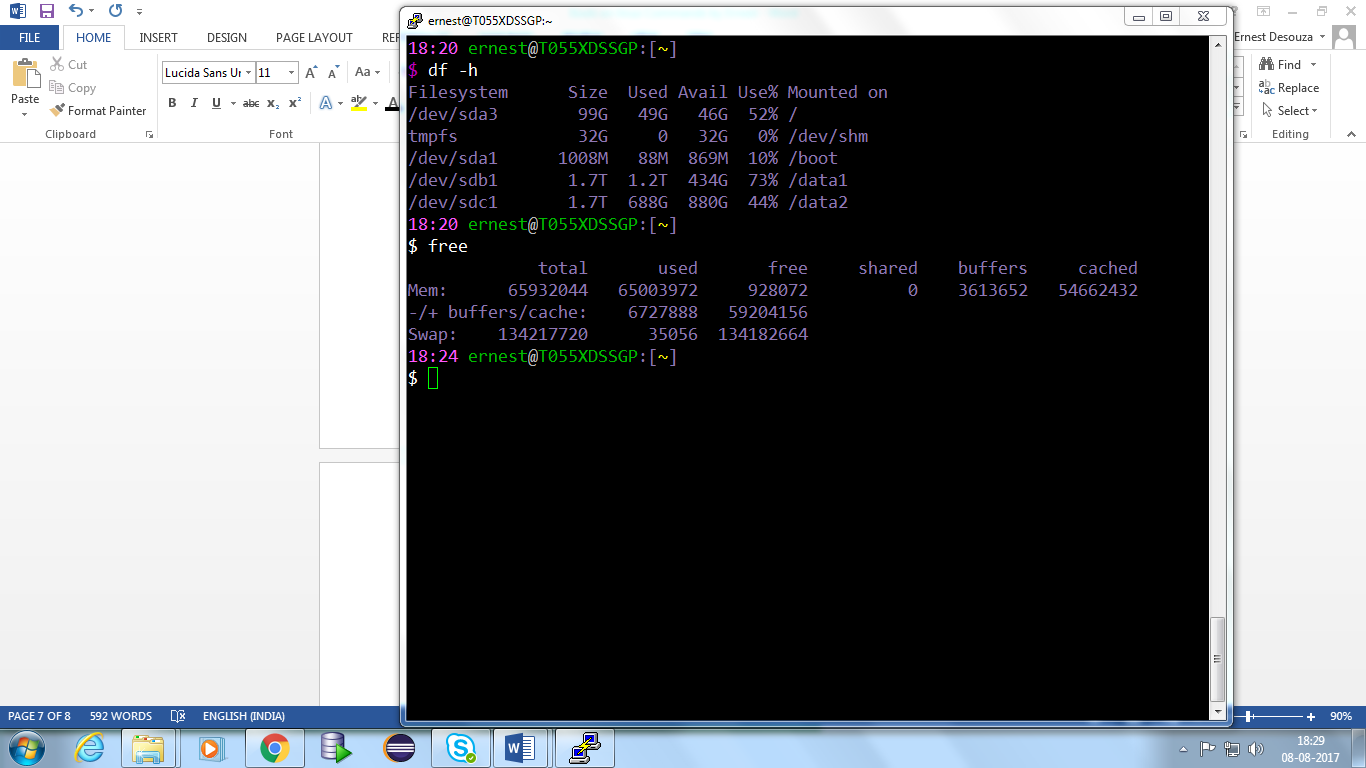
1. df-  How much space is left on a drive.

Syntax- df <options>



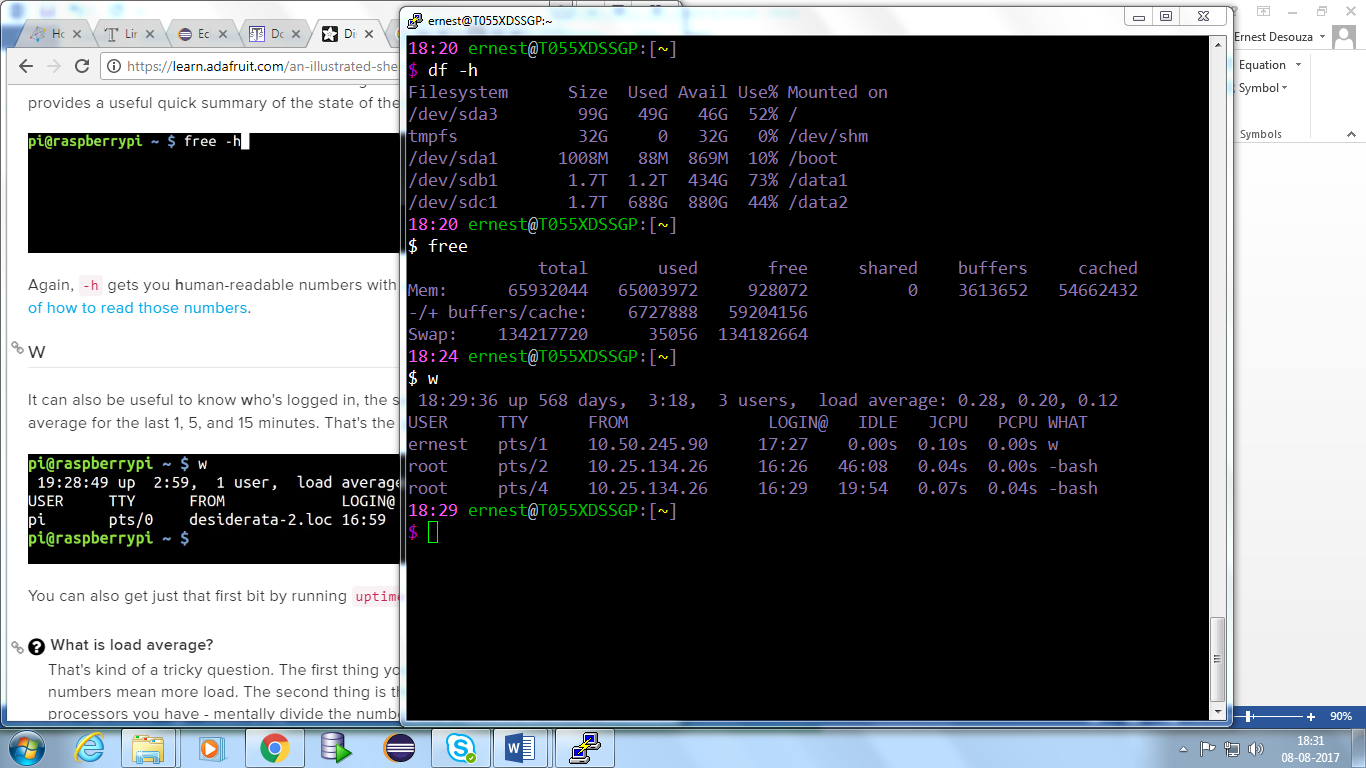
1. free- Provides a useful quick summary of the state of the computer's memory(RAM)

Syntax:free <option>



1. w- It can also be useful to know **w**ho's logged in, the system's uptime, and CPU load average for the last 1, 5, and 15 minutes.

Syntax- w



**Shell scripting (./bashrc)**

17:30 ernest@T055XDSSGP:[~]

$ nano .bashrc

GNU nano 2.0.9 File: .bashrc

echo=”This is my PC dare anyone touches it…..”

# Reset

Color\_Off='\[\e[0m\]' # Text Reset

# Colors will work nice when used with "Solarized" palette.

# Nothing new here

White='\[\e[0;37m\]' # White

BPurple='\[\e[1;35m\]' # Purple

Green='\[\e[0;32m\]' # Green

Blue='\[\e[0;34m\]' # Blue

Yellow='\[\e[0;33m\]' # Yellow

Purple='\[\e[0;35m\]' # Purple

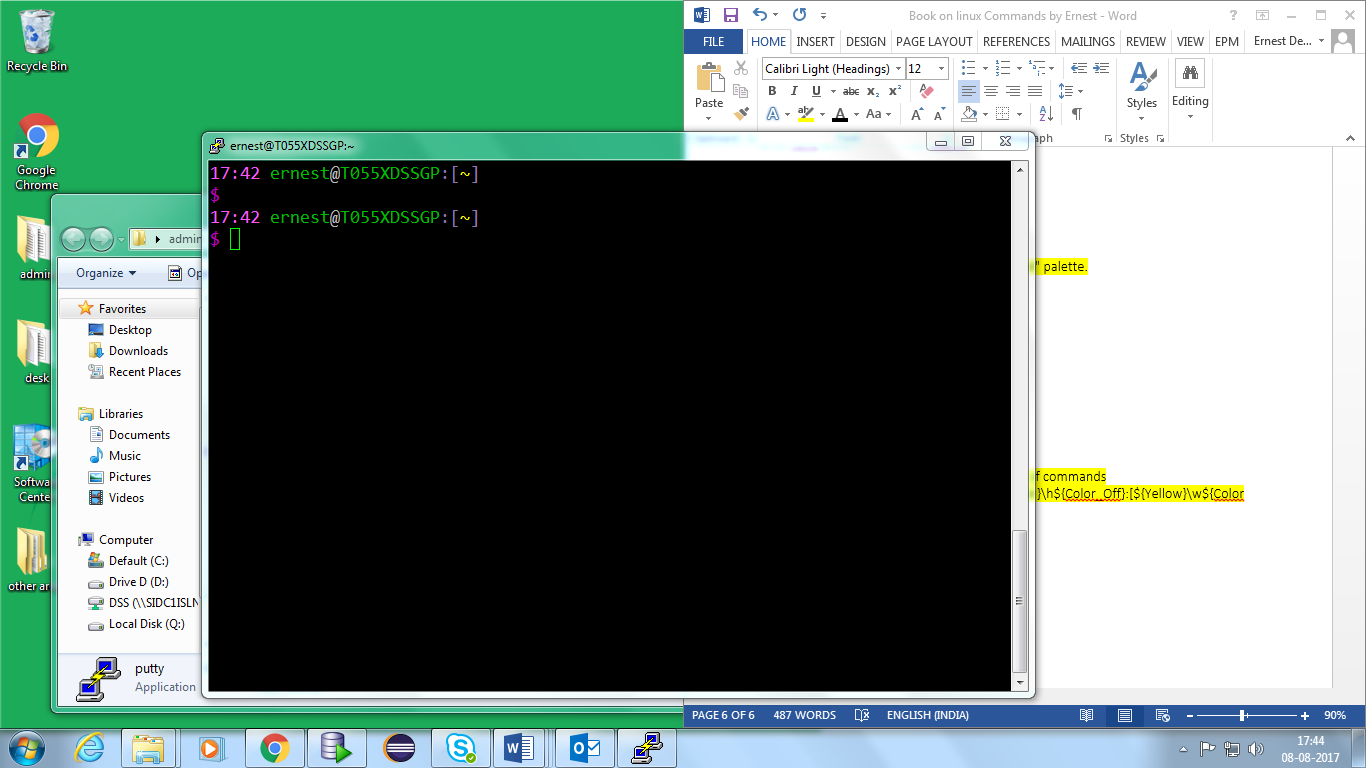
BWhite='\[\e[1;37m\]' # White

# custom prompt.

# User input is colored in white

# the trap will reset the colors before execution of Syntax

PS1="${BPurple}\A ${Green}\u${White}@${Green}\h${Color\_Off}:[${Yellow}\w${Color \_Off}]${Purple}\n$ ${BWhite}"



Note (what is new) -->

1. Added date.
2. Changed color of the variables manually.
3. Insert command on the next line.
4. Welcome note