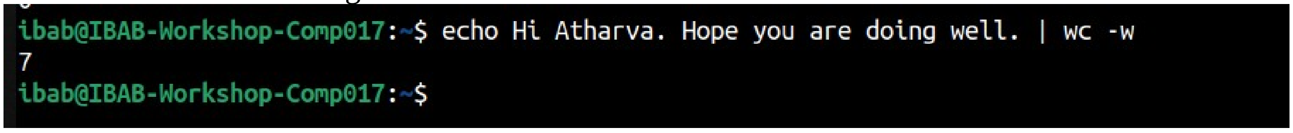
# Lab 6

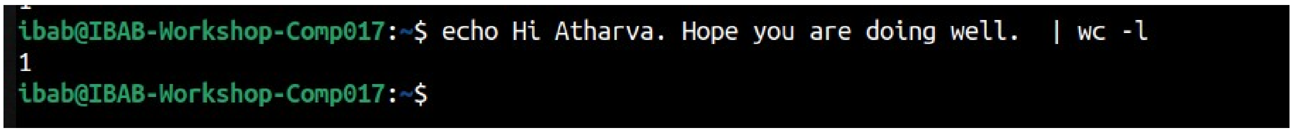
(1) The echo command can be used simply to display a piece of text (useful in programs). Use echo to create a sentence. Pipe the output of this command to count the number of words in the sentence. How do I use the wc to count the number of lines in this sentence?

The ***wc -w*** command will give the count of words in a sentence.



```
ibab@IBAB-Workshop-Comp017:~$ echo Hi Atharva. Hope you are doing well. | wc -w
7
ibab@IBAB-Workshop-Comp017:~$
```

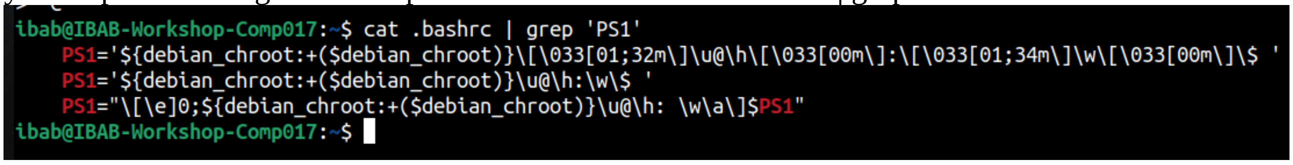
.



```
ibab@IBAB-Workshop-Comp017:~$ echo Hi Atharva. Hope you are doing well.  | wc -l
1
ibab@IBAB-Workshop-Comp017:~$
```

For printing lines use the command ***wc -l*** it will p[rint the numbber of lines in thie sentence.

(2) Execute the following command, and explain the meaning of the output. This concerns the appearance of the command prompt, and you should be able to dissect the output completely and explain it part by part. Make a list of the parts of the output and write your explanations against each part. The command is cat .bashrc | grep 'PS1'



```
ibab@IBAB-Workshop-Comp017:~$ cat .bashrc | grep 'PS1'
    PS1='${debian_chroot:+($debian_chroot)}\[\033[01;32m\]\u@\h\[\033[00m\]:\[\033[01;34m\]\w\[\033[00m\]\$ '
    PS1='${debian_chroot:+($debian_chroot)}\u@\h:\w\$ '
    PS1="\[\e]0;${debian_chroot:+($debian_chroot)}\u@\h: \w\a\]$PS1"
ibab@IBAB-Workshop-Comp017:~$ 
```

.

***PS1='${debian_chroot:+($debian_chroot)}\[\033[01;32m\]\u@\h\[\033[00m\]:\[\033[01;34m\]\w\[\033[00m\]\$ '*** -

this command changes the ***command prompts's font to debian_chroot***

and it changes the command proimpt to ***green colour*** becuase of the colour code ***32m*** given . And because of 1 before the colour code the prompt looks **bold.** It gives no colour i.e white to the colon because no code is given. While the "~" symbol is **blue** because of the code **34m** .

\u – stands for user

\h -stands for hostname

\w-it is a command line utility used to display information about the users

***PS1='${debian_chroot:+($debian_chroot)}\u@\h:\w\$ '***

It changes the command prompt to white and normal text as no colour codes are given.

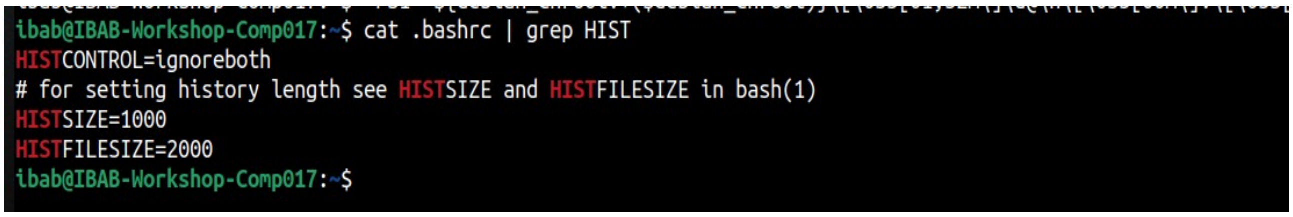
***PS1="\[\e]0;${debian_chroot:+($debian_chroot)}\u@\h: \w\a\]$PS1"***

It gives the terminal a title and the /w meaans the current working.

u- for username

h- for hostname

(3) In the above exercise, filter the output based on 'HIST' pattern. You will see a list of environment variables. Figure out what they stand for and what the current values mean.

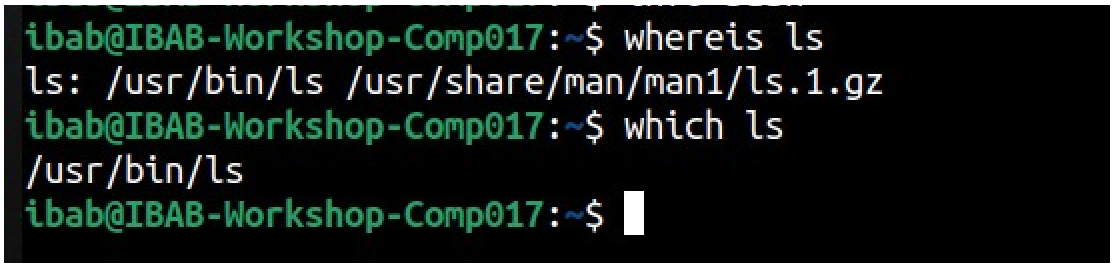


```
ibab@IBAB-Workshop-Comp017:~$ cat .bashrc | grep HIST
HISTCONTROL=ignoreboth
# for setting history length see HISTSIZE and HISTFILESIZE in bash(1)
HISTSIZE=1000
HISTFILESIZE=2000
ibab@IBAB-Workshop-Comp017:~$
```

**HISTCONTROL**= It saves commands in history according to the conditions we guve so we can control which commands should be saved and which shouldnt .
**Ignoreboth** means shorthand for **ignorespace** and **ignoredups** so all the commands willl be stored which have space and dups.
**HISTSIZE** = The number of commands to remember in the command history (see HISTORY below). If the value is 0, commands are not saved in the history list. Numeric values less than zero result in every command being saved on the history list (there is no limit). The shell sets the default value to 500 after reading any startup files. **But here we have the value set to 1000 so history will save 1000 commands.**

**HISTFILESIZE**
The maximum number of lines contained in the history file. When this variable is assigned a value, the history file is shortened, if necessary, to contain no more than that number of lines by removing the oldest entries. The history file is also shortened to this size after writing it when a shell exits. If the value is 0, the history file is truncated to zero size. Non-numeric values and numeric values less than zero inhibit truncation. The shell sets the default value to the value of HISTSIZE after reading any startup files.
**Here the size was set to 2000 so it will contain maximum of 2000 lines.**

(4) The whereis command searches for a program in a predefined set of standard binary directories such as /bin, /usr/bin and /usr/sbin. Type whereis ls and study the output. What is the difference between which and whereis commands?

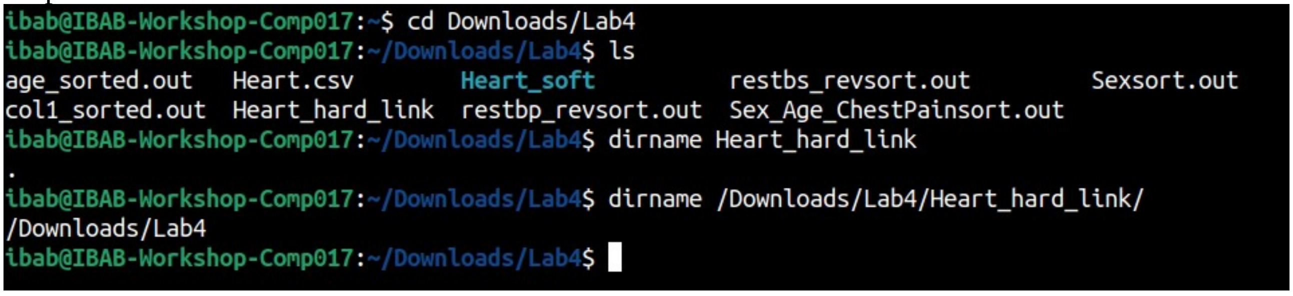


```
ibab@IBAB-Workshop-Comp017:~$ whereis ls
ls: /usr/bin/ls /usr/share/man/man1/ls.1.gz
ibab@IBAB-Workshop-Comp017:~$ which ls
/usr/bin/ls
ibab@IBAB-Workshop-Comp017:~$
```

The which command tells where the command is being executed and gives the path of the location where it is being executed.

The whereis command locates all the locations where the desired program is stored in the standard Linux places while which commands only lists the executed path of the command

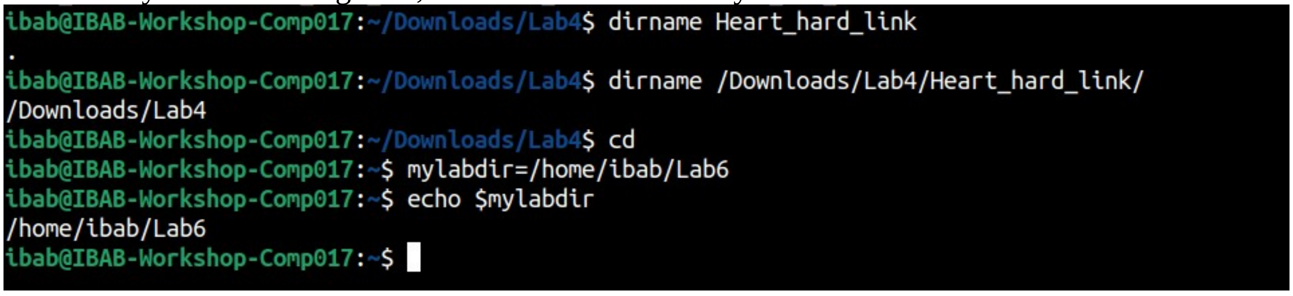
(5) The command dirname is used to retrieve the directory name in a given file path. Navigate to a previous lab folder such as Lab4 and list the files. Then type the command dirname <filename> where give some existing filename in this command. What is the output?



```
ibab@IBAB-Workshop-Comp017:~$ cd Downloads/Lab4
ibab@IBAB-Workshop-Comp017:~/Downloads/Lab4$ ls
age_sorted.out   Heart.csv        Heart_soft          restbs_revsort.out         Sexsort.out
col1_sorted.out  Heart_hard_link  restbp_revsort.out  Sex_Age_ChestPainsort.out
ibab@IBAB-Workshop-Comp017:~/Downloads/Lab4$ dirname Heart_hard_link
.
ibab@IBAB-Workshop-Comp017:~/Downloads/Lab4$ dirname /Downloads/Lab4/Heart_hard_link/
/Downloads/Lab4
ibab@IBAB-Workshop-Comp017:~/Downloads/Lab4$
```

.

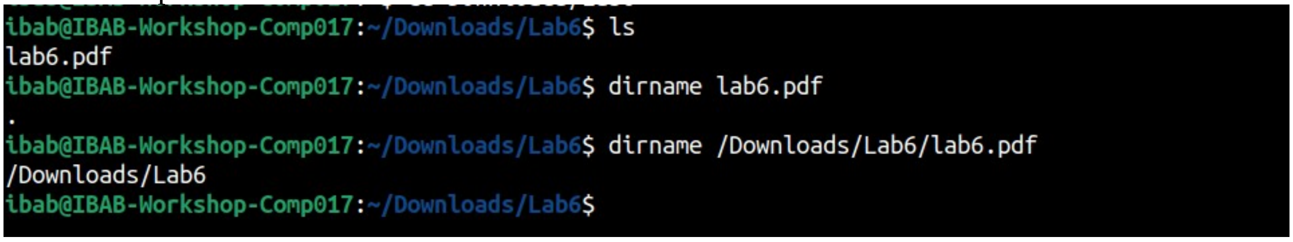
(6) Create a local variable called mylabdir, and set it to the following value: /home/ibab/Lab6 (which means that if you have not created a Lab6 folder you should do so). Print the value of this variable using the echo command, and then the dirname command output with a file you are working with, under the Lab6 directory.



```
ibab@IBAB-Workshop-Comp017:~/Downloads/Lab4$ dirname Heart_hard_link
.
ibab@IBAB-Workshop-Comp017:~/Downloads/Lab4$ dirname /Downloads/Lab4/Heart_hard_link/
/Downloads/Lab4
ibab@IBAB-Workshop-Comp017:~/Downloads/Lab4$ cd
ibab@IBAB-Workshop-Comp017:~$ mylabdir=/home/ibab/Lab6
ibab@IBAB-Workshop-Comp017:~$ echo $mylabdir
/home/ibab/Lab6
ibab@IBAB-Workshop-Comp017:~$
```

.

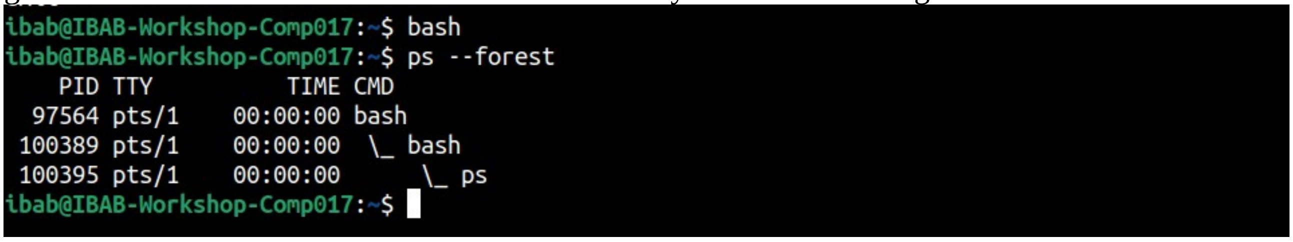
dirname output:



```
ibab@IBAB-Workshop-Comp017:~/Downloads/Lab6$ ls
lab6.pdf
ibab@IBAB-Workshop-Comp017:~/Downloads/Lab6$ dirname lab6.pdf
.
ibab@IBAB-Workshop-Comp017:~/Downloads/Lab6$ dirname /Downloads/Lab6/lab6.pdf
/Downloads/Lab6
ibab@IBAB-Workshop-Comp017:~/Downloads/Lab6$
```

.

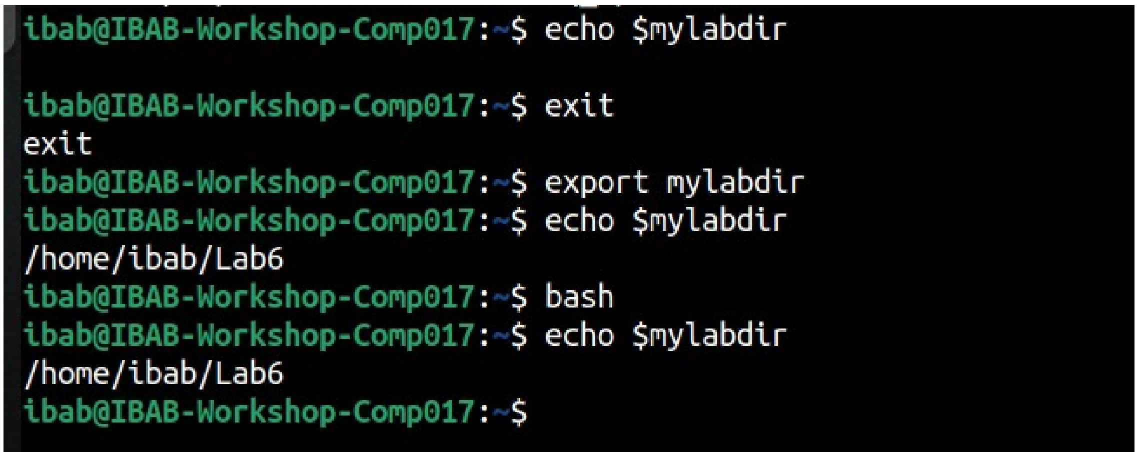
(7) Create a new bash subshell in this terminal. Prove that you have created a subshell using the ps command with the appropriate options. Inside this subshell print the value of the variable mylabdir. What is the value? If it is empty, how do you convert it to a global variable? Do the conversion and show that mylabdir is indeed a global variable.



```
ibab@IBAB-Workshop-Comp017:~$ bash
ibab@IBAB-Workshop-Comp017:~$ ps --forest
    PID TTY          TIME CMD
  97564 pts/1    00:00:00 bash
 100389 pts/1    00:00:00  \_ bash
 100395 pts/1    00:00:00      \_ ps
ibab@IBAB-Workshop-Comp017:~$
```

.

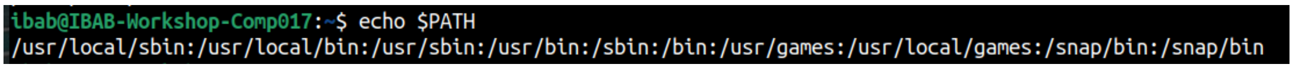
We can see that a sub shell has been created.



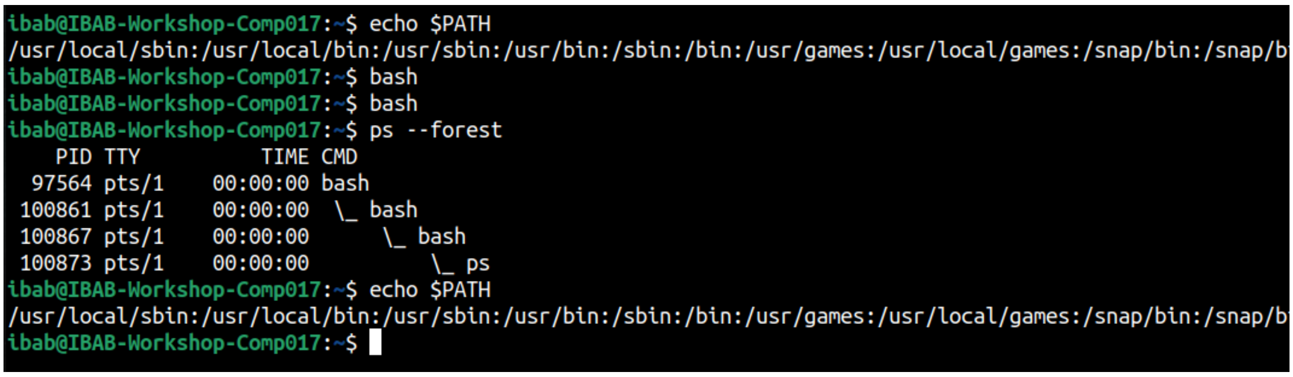
```
ibab@IBAB-Workshop-Comp017:~$ echo $mylabdir

ibab@IBAB-Workshop-Comp017:~$ exit
exit
ibab@IBAB-Workshop-Comp017:~$ export mylabdir
ibab@IBAB-Workshop-Comp017:~$ echo $mylabdir
/home/ibab/Lab6
ibab@IBAB-Workshop-Comp017:~$ bash
ibab@IBAB-Workshop-Comp017:~$ echo $mylabdir
/home/ibab/Lab6
ibab@IBAB-Workshop-Comp017:~$
```

.
We exported the variable and thus now we can access it in other sub shells also.

(8) Display the output of the command echo $PATH. Is PATH local or global variable? Describe two ways of finding this out.

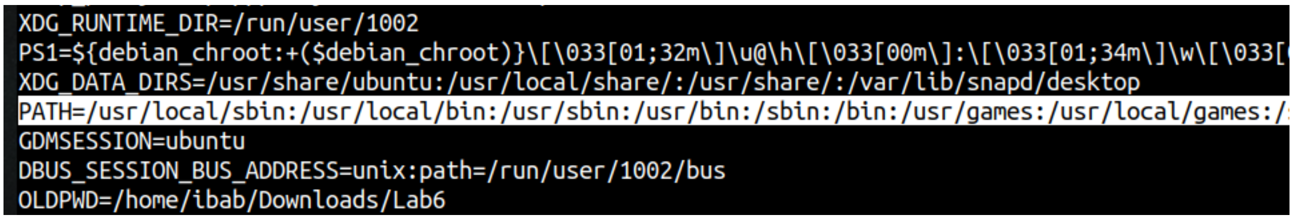


```
ibab@IBAB-Workshop-Comp017:~$ echo $PATH
/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin:/usr/games:/usr/local/games:/snap/bin:/snap/bin
```



```
ibab@IBAB-Workshop-Comp017:~$ echo $PATH
/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin:/usr/games:/usr/local/games:/snap/bin:/snap/b
ibab@IBAB-Workshop-Comp017:~$ bash
ibab@IBAB-Workshop-Comp017:~$ bash
ibab@IBAB-Workshop-Comp017:~$ ps --forest
    PID TTY          TIME CMD
  97564 pts/1    00:00:00 bash
 100861 pts/1    00:00:00  \_ bash
 100867 pts/1    00:00:00      \_ bash
 100873 pts/1    00:00:00          \_ ps
ibab@IBAB-Workshop-Comp017:~$ echo $PATH
/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin:/usr/games:/usr/local/games:/snap/bin:/snap/b
ibab@IBAB-Workshop-Comp017:~$ 
```

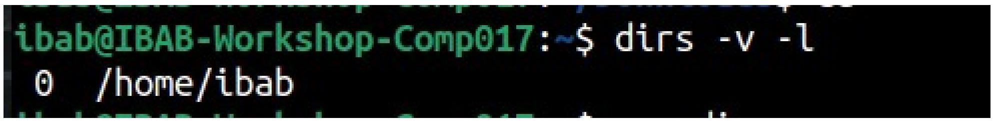
PATH is a Global environment variable.  One way of seeing that is by printing ***PATH*** in parent shell and a child sub-shell.



```
XDG_RUNTIME_DIR=/run/user/1002
PS1=${debian_chroot:+($debian_chroot)}\[\033[01;32m\]\u@\h\[\033[00m\]:\[\033[01;34m\]\w\[\033[
XDG_DATA_DIRS=/usr/share/ubuntu:/usr/local/share/:/usr/share/:/var/lib/snapd/desktop
PATH=/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin:/usr/games:/usr/local/games:/
GDMSESSION=ubuntu
DBUS_SESSION_BUS_ADDRESS=unix:path=/run/user/1002/bus
OLDPWD=/home/ibab/Downloads/Lab6
```

Second Way is by seeing in ***env*** command which lists all the global variables.

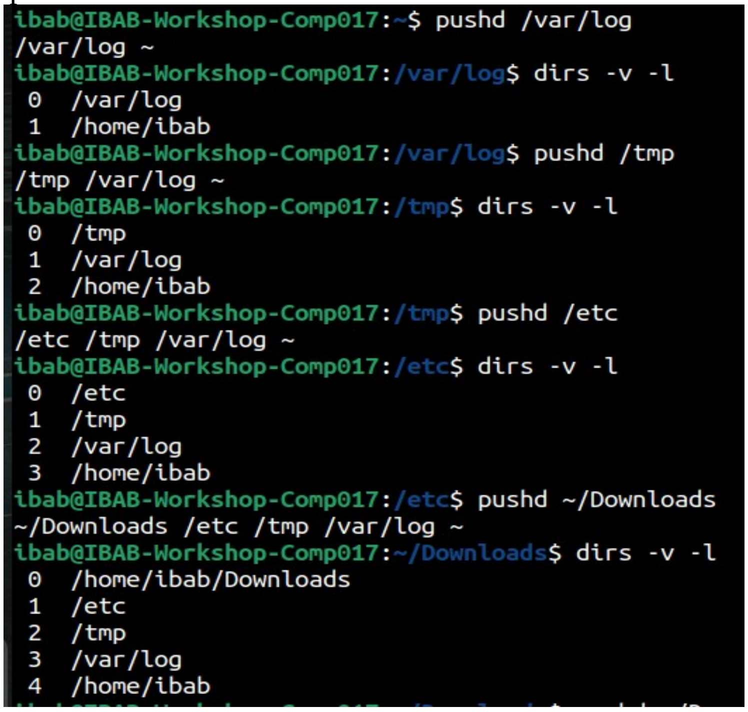
(9) The concept of a directory stack. Learning about this will allow you to know the powerful navigation mechanisms in Linux. The key commands in this exercise are dirs,cd -, pushd and popd. The directory stack operates like a stack of plates – the last plate on top is the first to be taken out. Try the following exercises to understand this.
(i) Execute dirs -v -l. What is the output? Learn more about the command and the options by looking up the man pages of what these options did. For each command below execute dirs -v -l to understand what happened to the directory stack.



```
ibab@IBAB-Workshop-Comp017:~$ dirs -v -l
 0  /home/ibab
```

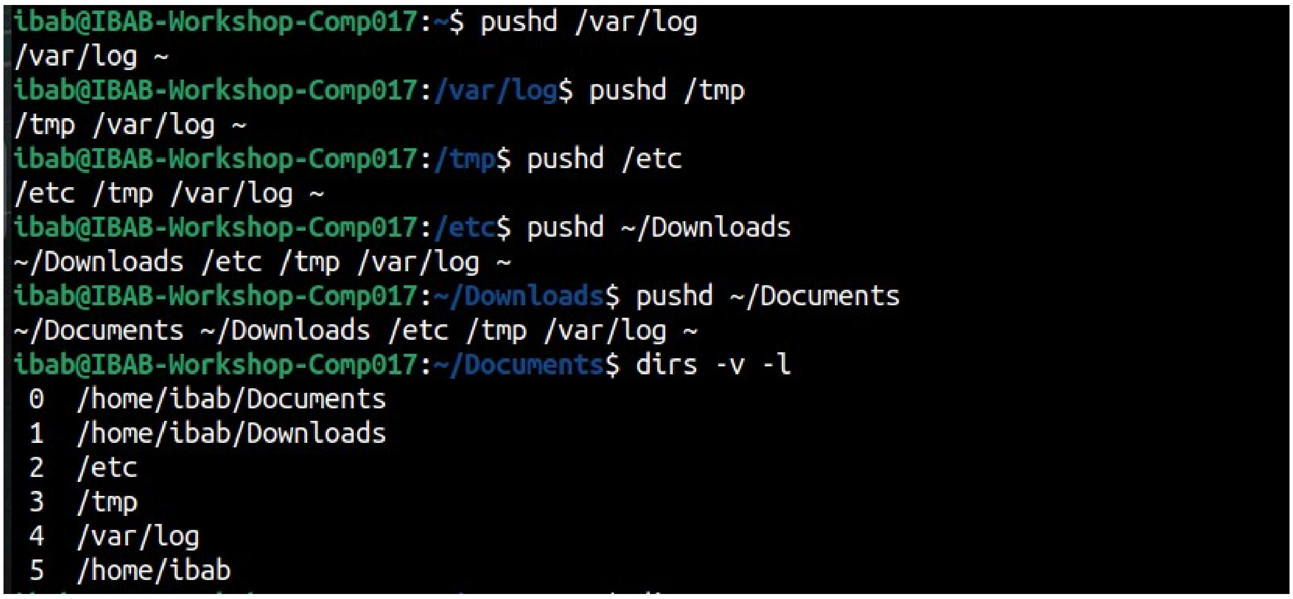
-s option prints the allocated size of each file, in blocks
-v option naturally sorts (version) numbers within text
-l option gives the long listing format

(ii) By default the directory stack contains only one entry – the path to your $HOME.
Let’s add some entries to this stack using the following commands:
(a) pushd /var/log
(b) pushd /tmp
(c) pushd /etc
(d) pushd ~/Downloads
(e) pushd ~/Documents



```
ibab@IBAB-Workshop-Comp017:~$ pushd /var/log
/var/log ~
ibab@IBAB-Workshop-Comp017:/var/log$ dirs -v -l
 0  /var/log
 1  /home/ibab
ibab@IBAB-Workshop-Comp017:/var/log$ pushd /tmp
/tmp /var/log ~
ibab@IBAB-Workshop-Comp017:/tmp$ dirs -v -l
 0  /tmp
 1  /var/log
 2  /home/ibab
ibab@IBAB-Workshop-Comp017:/tmp$ pushd /etc
/etc /tmp /var/log ~
ibab@IBAB-Workshop-Comp017:/etc$ dirs -v -l
 0  /etc
 1  /tmp
 2  /var/log
 3  /home/ibab
ibab@IBAB-Workshop-Comp017:/etc$ pushd ~/Downloads
~/Downloads /etc /tmp /var/log ~
ibab@IBAB-Workshop-Comp017:~/Downloads$ dirs -v -l
 0  /home/ibab/Downloads
 1  /etc
 2  /tmp
 3  /var/log
 4  /home/ibab
```

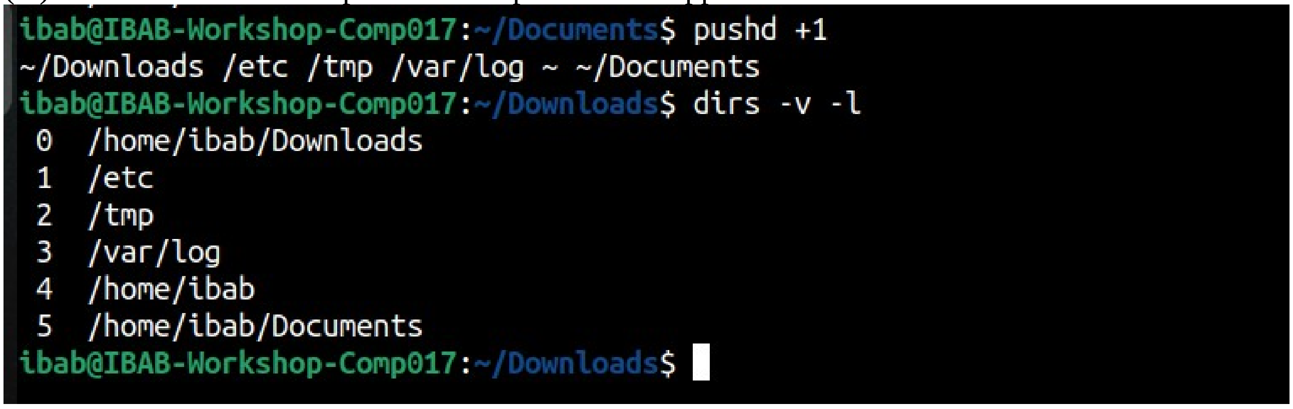
(iii)Now execute dirs -v -l again. Explain the output.



```
ibab@IBAB-Workshop-Comp017:~$ pushd /var/log
/var/log ~
ibab@IBAB-Workshop-Comp017:/var/log$ pushd /tmp
/tmp /var/log ~
ibab@IBAB-Workshop-Comp017:/tmp$ pushd /etc
/etc /tmp /var/log ~
ibab@IBAB-Workshop-Comp017:/etc$ pushd ~/Downloads
~/Downloads /etc /tmp /var/log ~
ibab@IBAB-Workshop-Comp017:~/Downloads$ pushd ~/Documents
~/Documents ~/Downloads /etc /tmp /var/log ~
ibab@IBAB-Workshop-Comp017:~/Documents$ dirs -v -l
 0  /home/ibab/Documents
 1  /home/ibab/Downloads
 2  /etc
 3  /tmp
 4  /var/log
 5  /home/ibab
```

Here the latest directories which are being stacked are given, the serial number 0 and likewise lists the directories from the latest to the oldest and also the path in the long format has been given.

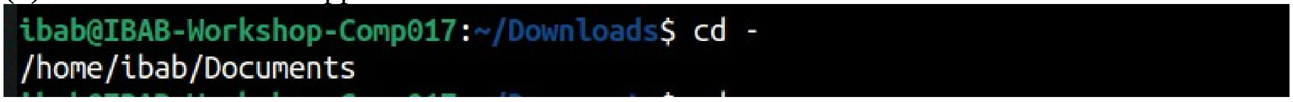
(iv)Execute the command pushd +1. Explain what happened.



```
ibab@IBAB-Workshop-Comp017:~/Documents$ pushd +1
~/Downloads /etc /tmp /var/log ~ ~/Documents
ibab@IBAB-Workshop-Comp017:~/Downloads$ dirs -v -l
 0  /home/ibab/Downloads
 1  /etc
 2  /tmp
 3  /var/log
 4  /home/ibab
 5  /home/ibab/Documents
ibab@IBAB-Workshop-Comp017:~/Downloads$
```

So, the current directory i.e. Downloads shifted one position up and and it became the current working directory while the directory which was on 0th position went to the last position it kind of rotated.

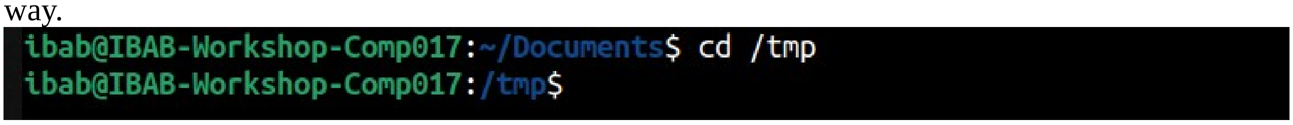
(v)Execute cd -. What happened?



```
ibab@IBAB-Workshop-Comp017:~/Downloads$ cd -
/home/ibab/Documents
```

It gave the absolute path to the previous directory it is used to navigate to the previous directory.

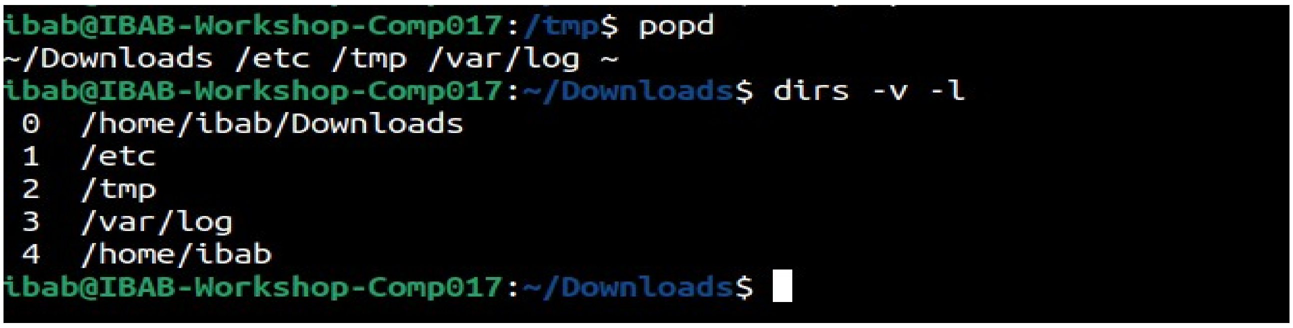
(vi)Execute cd /tmp. What happened? In this case, note that we simply changed directory to an entry in the directory stack without reference to the stack in any way.



```
ibab@IBAB-Workshop-Comp017:~/Documents$ cd /tmp
ibab@IBAB-Workshop-Comp017:/tmp$
```

The directory got changed ***tmp***

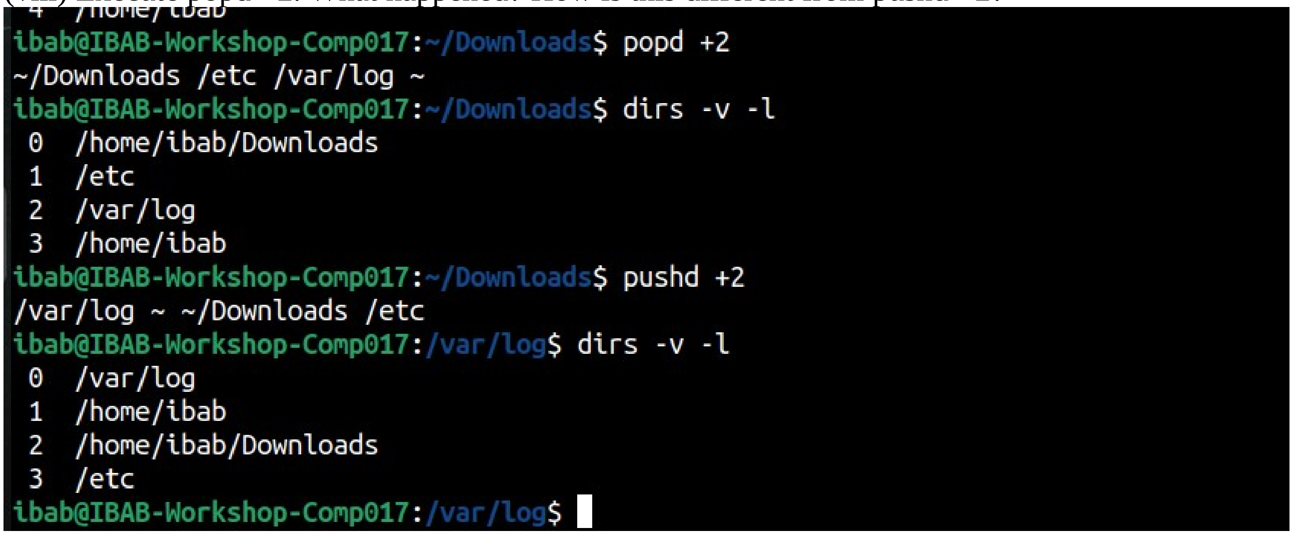
(vii) Execute popd. What happened? popd is responsible for removing the topmost "plate" in the stack and changes to the directory entry against index 1.



```
ibab@IBAB-Workshop-Comp017:/tmp$ popd
~/Downloads /etc /tmp /var/log ~
ibab@IBAB-Workshop-Comp017:~/Downloads$ dirs -v -l
 0  /home/ibab/Downloads
 1  /etc
 2  /tmp
 3  /var/log
 4  /home/ibab
ibab@IBAB-Workshop-Comp017:~/Downloads$
```

The topmost directory here ***Documents*** was removed and it was replaced by ***Downloads*** which was at position 1

(viii) Execute popd +2. What happened? How is this different from pushd +2?



```
 4  /home/ibab
ibab@IBAB-Workshop-Comp017:~/Downloads$ popd +2
~/Downloads /etc /var/log ~
ibab@IBAB-Workshop-Comp017:~/Downloads$ dirs -v -l
 0  /home/ibab/Downloads
 1  /etc
 2  /var/log
 3  /home/ibab
ibab@IBAB-Workshop-Comp017:~/Downloads$ pushd +2
/var/log ~ ~/Downloads /etc
ibab@IBAB-Workshop-Comp017:/var/log$ dirs -v -l
 0  /var/log
 1  /home/ibab
 2  /home/ibab/Downloads
 3  /etc
ibab@IBAB-Workshop-Comp017:/var/log$
```

In this, ***popd +2*** removes the directory at the 2nd position and shifts the rest of the directories 1 position up. While ***pushd +2*** doesnt remove any directory it simply rotates or pushes the directories 2 positions in the stack here, ***Downloads*** directory went up by 2 positions and the other directories also shifted 2 positions above.

(ix) In all the above, what was happening the entry that was indexed against '0' ?
Explain what this entry represents.

The entry that was against index 0 rotated / shifted to the bottom of the list or whereever the position was specified, in popd function it was removed as well . And accordingly directories in the whole stack moved up a few positions . And this entry represents the current directory you are working in. **Pushd** adds a directory to the 0th position and changes current working directory to the directory which got pushed on 0th position.
**Popd** it removes the topmost directory i.e. directory at the 0th position from the stack and changes the directory at 1 position to the 0th position and changes it to the current working directory.