OPERATING SYSTEMS – INTRODUCTION TO LINUX #2

OCTOBER 15, 2016

16CSCI03I - OPERATION SYSTEMS

By Mohamed El Sayed

Operating Systems – Introduction to Linux #2

Note: Before studying this lab you must refer to "Introduction to Linux" i.e. Lab 1.

What will we study within this lab?

We will be learning more about few more important Linux commands, in addition to few handy shortcuts for some commands. The objective of this lab is to get you even more familiar with Linux capabilities.

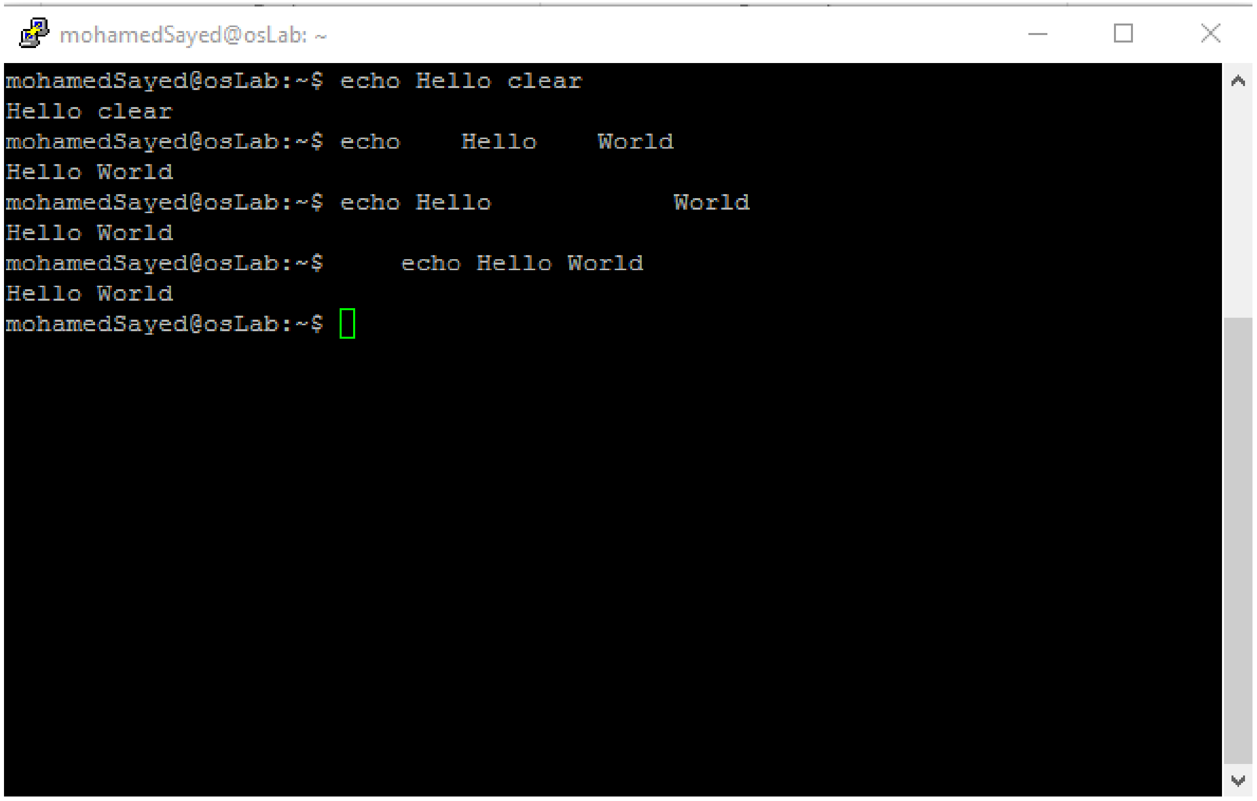
How does Shell deal with commands?

One of the primary features of a shell is to perform a command line scan. When you enter a command at the shell's command prompt and press the enter key, then the shell will start scanning that line, cutting it up in arguments. While scanning the line, the shell may make many changes to the arguments you typed.

This process is called **shell expansion**. When the shell has finished scanning and modifying that line, then it will be executed.

Shell dealing with White Spaces

Parts that are separated by one or more consecutive **white spaces** (or tabs) are considered separate **arguments**, any white space is removed. The first **argument** is the command to be executed, the other **arguments** are given to the command. The shell effectively cuts your command into one or more arguments. This explains why the following four different command lines are the same after shell expansion.



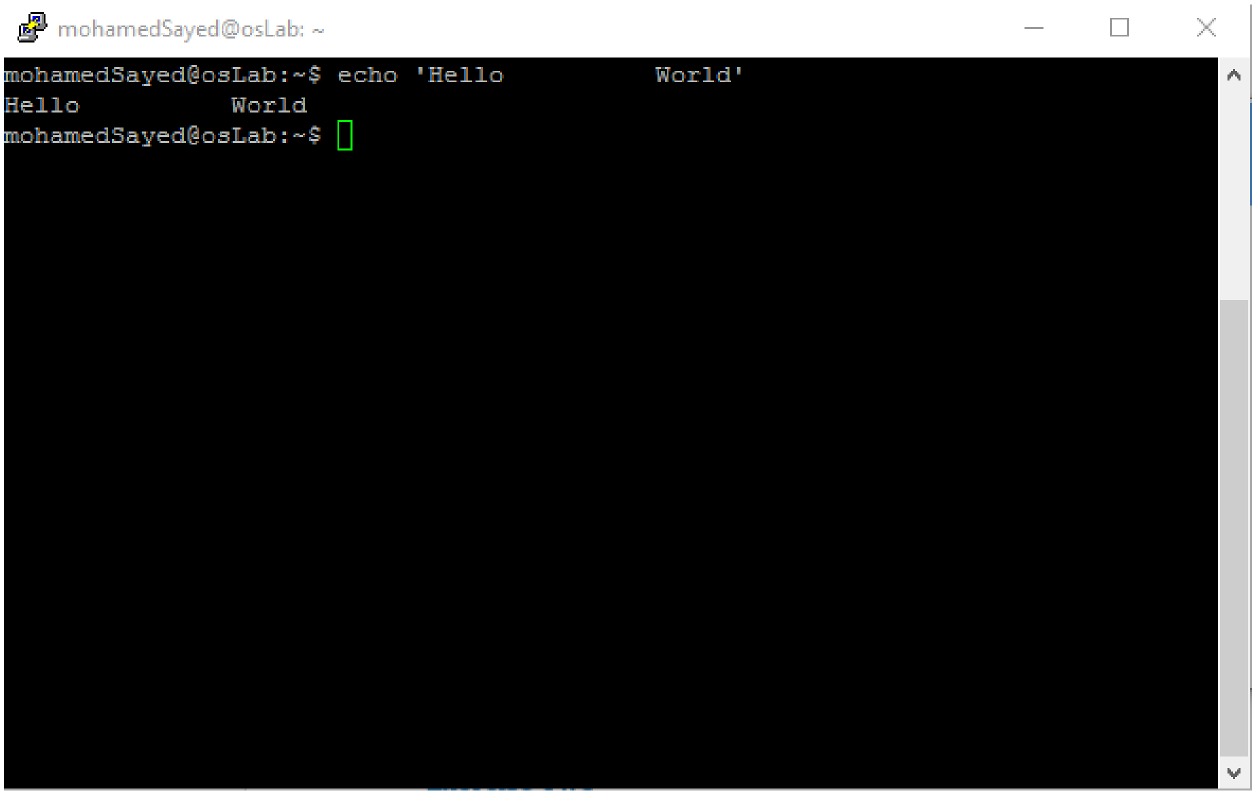
```
mohamedSayed@osLab:~$ echo Hello clear
Hello clear
mohamedSayed@osLab:~$ echo    Hello    World
Hello World
mohamedSayed@osLab:~$ echo Hello            World
Hello World
mohamedSayed@osLab:~$     echo Hello World
Hello World
mohamedSayed@osLab:~$ 
```

Figure 1. Showing shell expansion operation through dealing with white spaces.

Notice in above "echo Hello   World" example. The first argument "echo" sent to the shell that is the command to be executed. Whatever comes after it, in this example "Hello   World" are arguments which are sent to the command "echo" itself not the shell.

Single Quotes

You can prevent the removal of white spaces by quoting the spaces. The contents of the quoted string are considered as one argument. In the screenshot below the echo receives only one **argument**.



```
mohamedSayed@osLab:~$ echo 'Hello          World'
Hello          World
mohamedSayed@osLab:~$
```

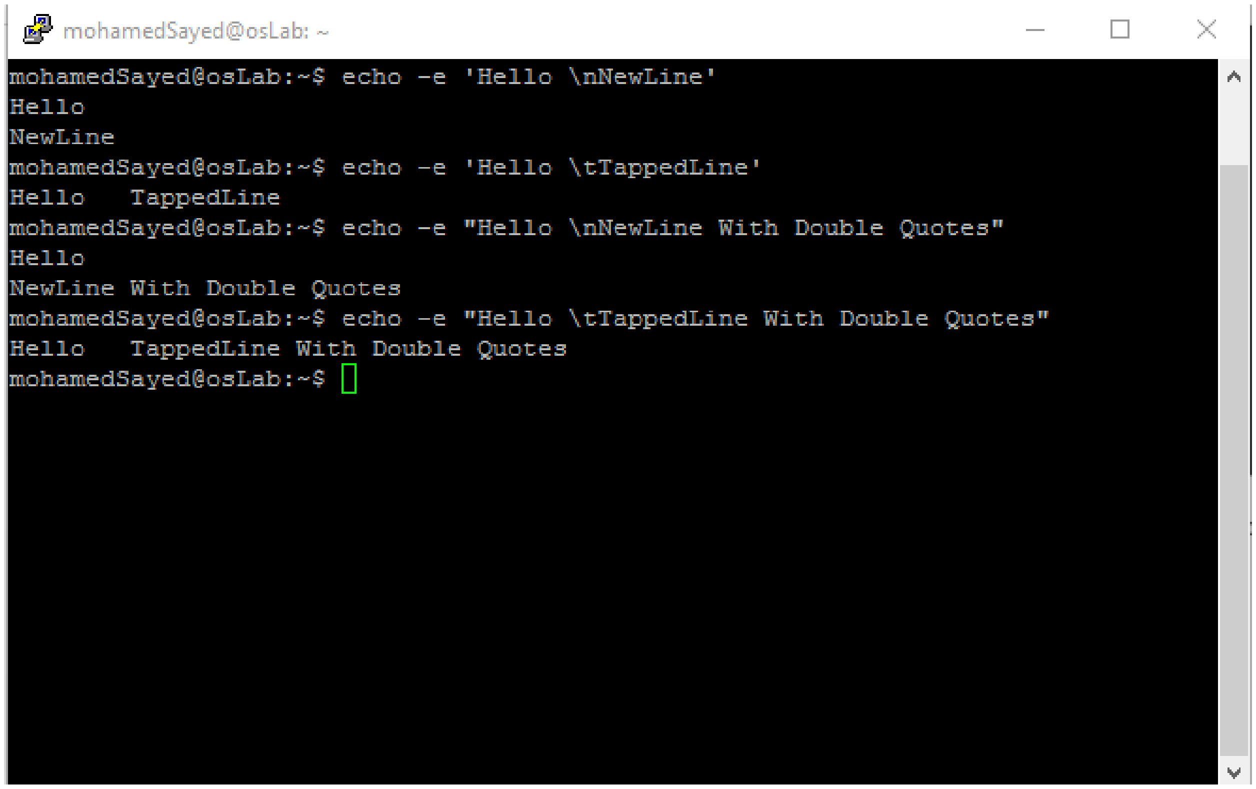
Figure 2. Showing echo's single quotes effect on white-spaces.

Double Quotes

You can also prevent the removal of white spaces by double quoting the spaces. Same as above, **echo** only receives one **argument**.

Echo and Quotes

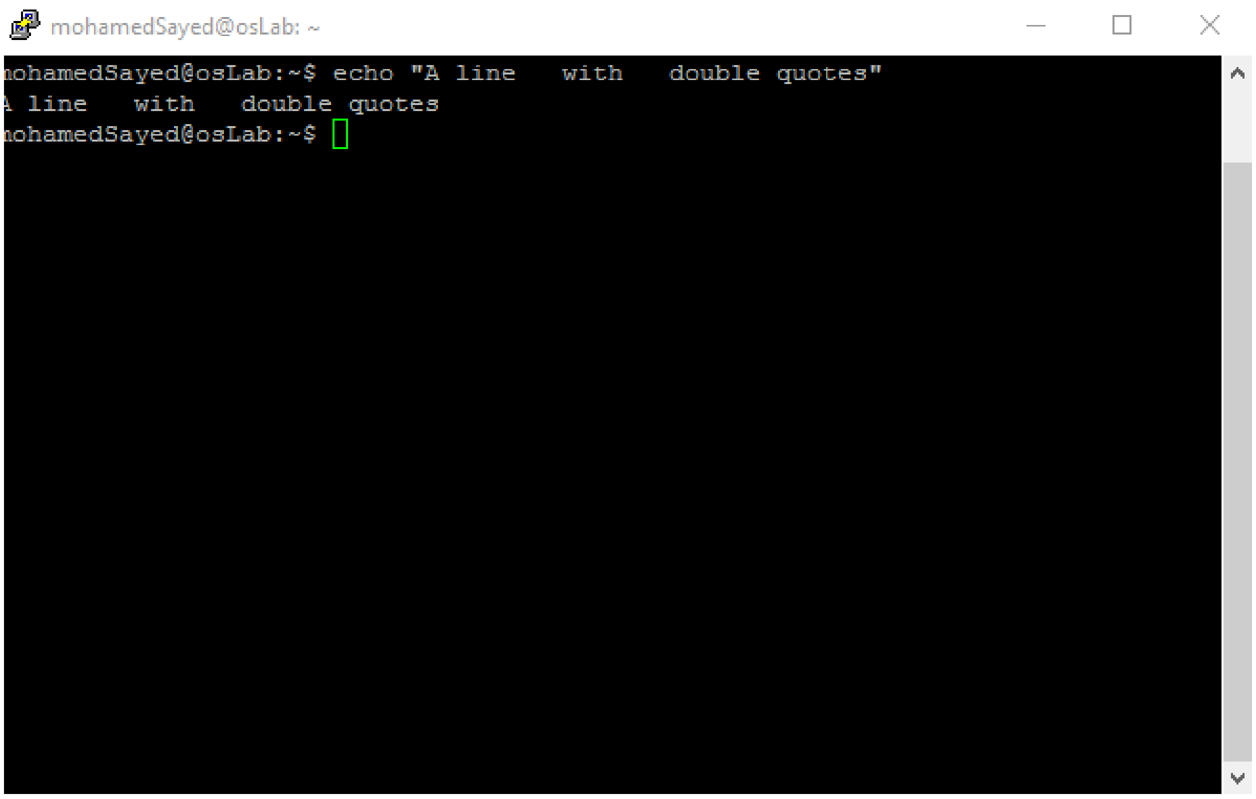
Quoted lines can include special escaped characters recognized by the echo command (when using **echo -e).** The screenshot below shows how to use **\n** for a newline and **\t** for a tab (usually eight white spaces).



```
mohamedSayed@osLab:~$ echo -e 'Hello \nNewLine'
Hello
NewLine
mohamedSayed@osLab:~$ echo -e 'Hello \tTappedLine'
Hello   TappedLine
mohamedSayed@osLab:~$ echo -e "Hello \nNewLine With Double Quotes"
Hello
NewLine With Double Quotes
mohamedSayed@osLab:~$ echo -e "Hello \tTappedLine With Double Quotes"
Hello   TappedLine With Double Quotes
mohamedSayed@osLab:~$ 
```

Figure 3. Showing echo’s quotes effect on special characters.

The echo command can generate more than white spaces, tabs and newlines. Look in the man page for a list of options.



```
ohamedSayed@osLab:~$ echo "A line   with   double quotes"
 line   with   double quotes
ohamedSayed@osLab:~$
```

Figure 3. Showing echo’s double quotes effect on white-spaces.

Commands Type

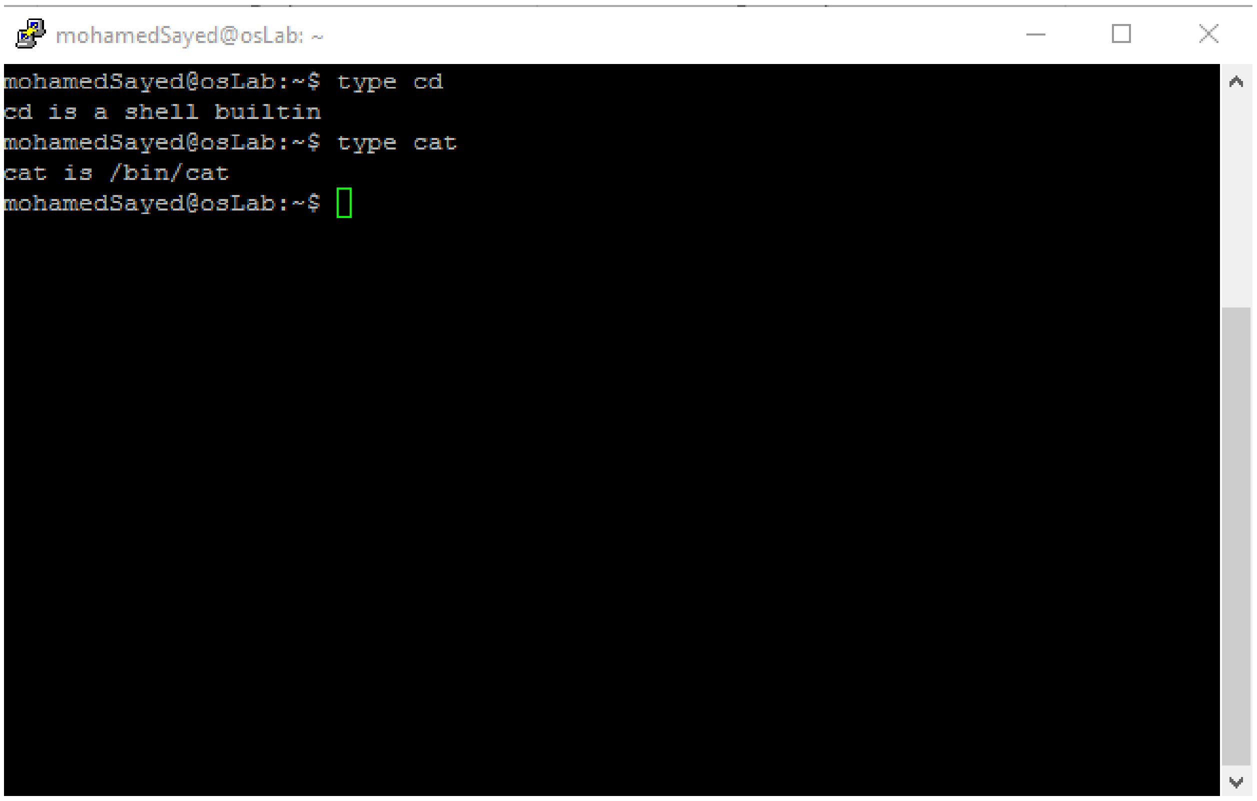
Linux is an open source Operating System which gives developers and it is community a wide space for contributing into the OS by adding commands, either locally on your Linux machine or globally. There may exists a command which does certain operation created by YOU on every Linux user’s machine! Those commands are defined as **external** commands, while the rest are builtin.

Builtin commands are an integral part of the shell program itself.

Type

To find out whether a command given to the shell will be executed as an **external command** or as a **builtin command**, use the **`type`** command.

```
type cd
type cat
```

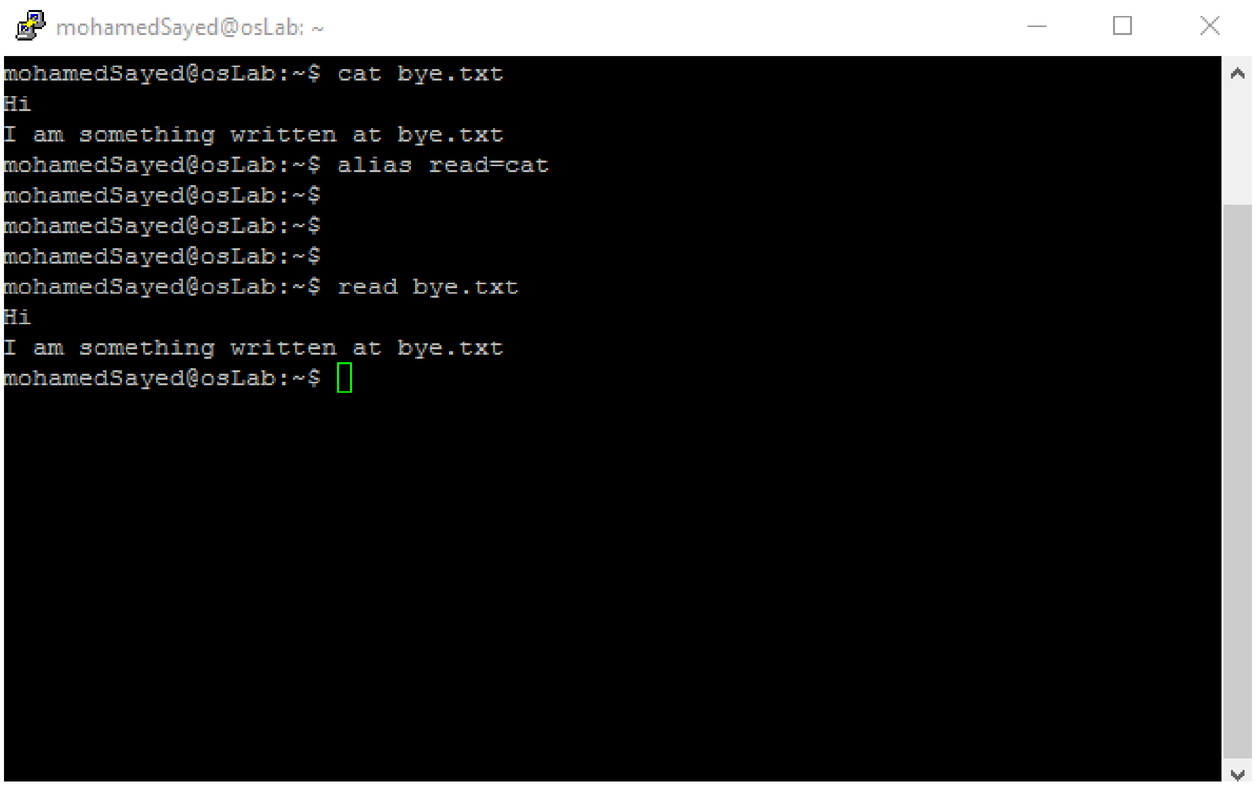


```
mohamedSayed@osLab:~$ type cd
cd is a shell builtin
mohamedSayed@osLab:~$ type cat
cat is /bin/cat
mohamedSayed@osLab:~$
```

Figure 4. Showing type command output.

Aliases – Create an alias

The shell allows you to create **aliases**. Aliases are often used to create an easier to remember name for an existing command or to easily supply parameters.



```
mohamedSayed@osLab:~$ cat bye.txt
Hi
I am something written at bye.txt
mohamedSayed@osLab:~$ alias read=cat
mohamedSayed@osLab:~$
mohamedSayed@osLab:~$
mohamedSayed@osLab:~$
mohamedSayed@osLab:~$ read bye.txt
Hi
I am something written at bye.txt
mohamedSayed@osLab:~$
```

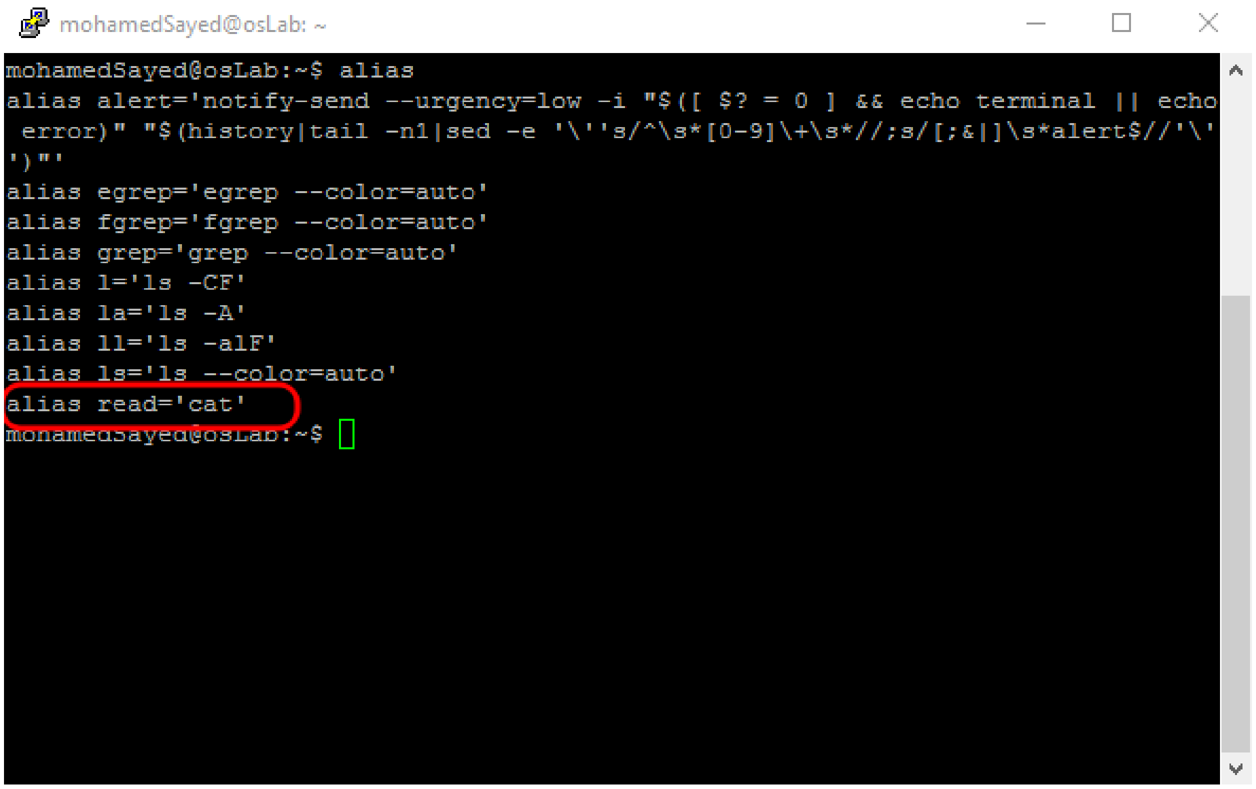
Figure 5. Showing alias command output.

In above example we have created an alias to the `cat` command. The alias is `read`, so whenever we execute `read`, the same `cat` operation will be done. You may think of an alias as a "user-preference" utility to set a "nickname" for any given command.

Aliases – Viewing aliases

By passing no arguments to the `alias` command, you are enforcing the behavior of "list me all the aliases on this machine."

```
alias
```



```
mohamedSayed@osLab:~$ alias
alias alert='notify-send --urgency=low -i "$([ $? = 0 ] && echo terminal || echo
 error)" "$(history|tail -n1|sed -e '\''s/^\s*[0-9]\+\s*//;s/[;&|]\s*alert$//'\'
')"'
alias egrep='egrep --color=auto'
alias fgrep='fgrep --color=auto'
alias grep='grep --color=auto'
alias l='ls -CF'
alias la='ls -A'
alias ll='ls -alF'
alias ls='ls --color=auto'
alias read='cat'
mohamedSayed@osLab:~$
```

Figure 6. Showing alias command with no arguments output.

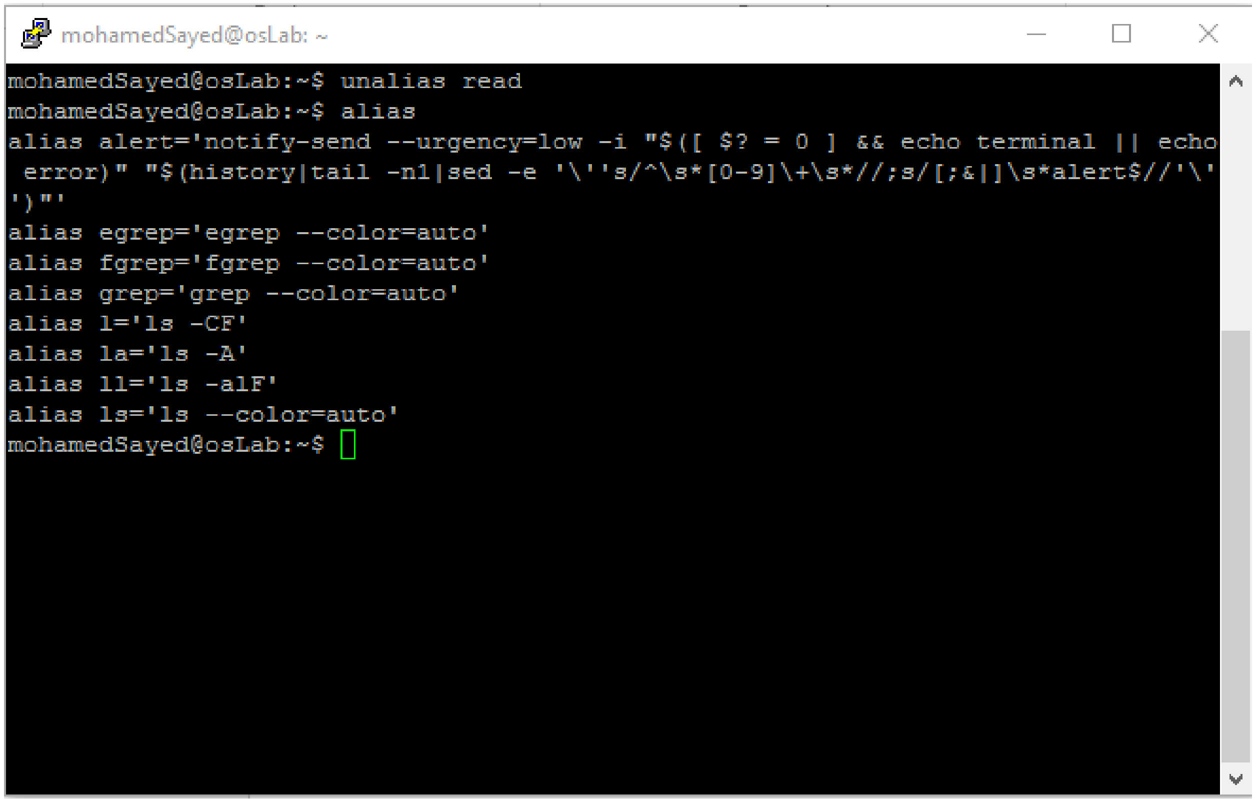
Notice the "read" alias we created for "cat" is listed at Figure 6.

Aliases – Deleting an alias (unalias)

You can undo an alias by executing the `unalias` command.

```
unalias read
```

In above example we have removed the alias created for `cat` that was `read`.



```
mohamedSayed@osLab:~$ unalias read
mohamedSayed@osLab:~$ alias
alias alert='notify-send --urgency=low -i "$([ $? = 0 ] && echo terminal || echo
 error)" "$(history|tail -n1|sed -e '\''s/^\s*[0-9]\+\s*//;s/[;&|]\s*alert$//'\'
')"'
alias egrep='egrep --color=auto'
alias fgrep='fgrep --color=auto'
alias grep='grep --color=auto'
alias l='ls -CF'
alias la='ls -A'
alias ll='ls -alF'
alias ls='ls --color=auto'
mohamedSayed@osLab:~$
```

Figure 7. Showing unalias command with no arguments output.

Notice the “read” alias no longer exists in Figure 7, unlike in Figure 6.

Handy Shortcuts – Bonus Tips for young Linux Geeks!

1. Have you made a typo within your command? Don’t write it again from scratch! Press the “Up” key on your keyboard and fix the typo (save time!).
2. Have you created a set of nested directories (directory1 which has directory2, which has directory3 inside, etc.)? You can directly navigate to directory3 within one `cd` operation, for instance, `cd directory1/directory2/directory3`
3. If you want to navigate back from directory3 to directory1 or even further before you may use `cd ../../../`
4. Would you like to navigate (cd) to a directory which has a VERY VERY long name? Don’t write it all down! Just type the first two or three letters and press the “Tab” key on your keyboard to trigger the autocomplete.

Exercise

1. How many arguments are in this line (not counting the command itself)?

```
touch '/etc/cron/cron.allow' 'file 42.txt' "file 33.txt"
```

2. Is `tac` a shell builtin command?
3. Is there an existing alias for `rm`?
4. Read the man page of `rm`, make sure you understand the `-i` option of `rm`. Create and remove a file to test the `-i` option.
5. Execute: `alias rm='rm -i'.` Test your alias with a test file. Does this work as expected?
6. List all current aliases.
7. Create an alias called 'city' that echoes your hometown.
8. Use your alias to test that it was created.
9. Remove your city alias.
10. What is the location of the `cat` and the `passwd` commands?
11. Explain the difference between the following commands `echo` and `/bin/echo`
12. Explain the difference between the following commands `echo Hello` and `echo -n Hello`
13. Display A B C with two spaces between B and C.
14. Use echo to display the following exactly

    ??\\

15. Use one echo command to display three words on three lines.