**Unix One liner examples**

**Advantages with one liner’s:**

* productivity
* Reduce system resource usage(In some cases )
* Short and sweet codes.

Command chaining operators

**& –Sends process background(run in background parallelly )**  
**; –Run multiple commands in one run, sequentially.**  
**\ –To combine a big command for multiple lines**  
**&& –Logical AND operator**  
**|| –Logical OR operator**  
**! -NOT operator(I did not find much info on this.0)**  
**| — PIPE operator**  
**{} –Command combination operator.**  
**() –Precedence operator**

**& – Runs a command in background(AND operator)**

This operator is used to send a process/script/command to run background so that we can execute other commands in foreground to increase effective utilization of system resources and to speed up the script execution. This is also called as Child process creation or forking in other programming languages.

**Example1:** Run commands in background

**$ping -c1 google.com &**

**Example2:** Run more commands in background in single line

**$ping -c1 google.com & scp root@1.1.1.10:/opt/\* /opt &**

Above commands are run in background parallelly independent of other commands. Like this we can run many such commands parallel.

**; – semicolon operator**

This operator Run multiple commands in one go but in a sequential order. So second command will run after first command completion, third command will run only after second command run.

**Example3:** Execute ls, pwd, whoami commands in one command

**ls;pwd;whoami**

**Note1:** The number of commands you can run is infinity. By default there is no limit on how many commands you can run with ; operator. We have checked this with 500 commands executed in one line. The limit depends only on memory or ulimits settings.

**Exampl4:** Run a bash shell script IF statement in one command

**if [ $? -eq 0 ]; then echo “sucess”; else echo “fail”; fi**

**Example5:** Run online FOR loop

**for i in `ls -1`;do echo “file is $i”;done**

**Example6:** Run online while loop

**while [ $VAR1 -eq 10 ]; do echo “Var value is $VAR1”; done**

**Example7:** Run online until loop

**until [ $VAR1 -eq 10 ]; do echo “Var value is $VAR1”; done**

This is a kind of shell scripting at terminal or one liner shell scripting or temporary shell scripting. This will help you not to create a file for your small and dirty scripts. Just execute your script directly at terminal.

**\ – Concatenation operator**

This operator is used to execute a command which is too big and spread on multiple lines. This operator helps to spread this command to do multi line execution.

**Example8:** Try to copy a file /var/ftp/pub/webadmin/debina/read from sysloggerserver1.linuxnix.com to /opt/oradba/debadmin/logic/abc folder on sysloggerserver2.persistentsystems.co.in and I am at surendrahome.beamtele.com.

**ssh abc**[**@sysloggerserver1.linuxnix.com**](mailto:root@sysloggerserver1.linuxnix.com)**: \**  
**/var/ftp/pub/webadmin/debian/read \**  
[**abc@sysloggerserver2.persistent.co.in**](mailto:root@sysloggerserver2.persistent.co.in)**\**  
**:/opt/oradba/debadmin/logic/abc/**

Actually this a big command but I divided this to multiple lines to make it look good and understandable by using .

**&& –Logical AND operator**

This operator lets you execute second command, if the first command runs successfully i.e [exit status](http://www.linuxnix.com/2011/03/find-exit-status-script-command-linux.html) is zero. This is very much useful to check the success status of first command. For example I want to connect to abc.com before connecting I want to check if that machine is pingable or not.

**Example9:** Connect to a machine only if its able to ping other wise dont execute connect command

**ping -c1 abc.com && ssh**[**abc@abc.com**](mailto:root@abc.com)

**Example10:** Assign a value to variable if it’s not set

**[ -z $VAR1 ] && VAR1=10**

**Example11:** Check if a folder exists or not. If it’s not present, create it.

**[ ! -d /var/temp ] && mkdir /var/temp**

**Example12:** Create folder, if the folder creation is successful then only change the directory.

**mkdir abc && cd abc**

**Example13:** Change to a folder, if this is success then list the content.

[**cd /var/ftp**](http://www.linuxnix.com/2012/07/23-linux-cd-command-examples.html)**&& ls**  
Like this you can use your imagination to use this operator effectively to save some time.

**|| –Logical OR operator**

This operator lets you execute second command, if the first command fails i.e exit status is greater than zero of first command. This is very much useful to check the failed status of first command. For example I want to create a folder, but before that I want to check if folder exits or not. If it’s not exists then create it.

**Example 14:** Create a folder if the folder does not exits.

**[ -d /var/temp ] || mkdir /var/temp**

**Example15:** Ping to a machine and let users know if it’s not reachable in a meaning full way.

**ping -c1 google.com**[**&> /dev/null**](http://www.linuxnix.com/2012/02/linuxunix-redirection-operatorsfile-descriptors-explained-examples.html)**|| echo “There is some problem with network, please connect your network admin”**

**&& and || operators combination**

We can use && and || to simulate if-else statement with in one line. If the first command executed successfully we can execute second command otherwise we will execute third command.

**Example16:** ping google.com, and display useful information to user if its pingable or not.

**ping -c1 google.com && echo “Thats good, able to ping google.com” || echo “Thats bad unable to ping google.com”**

**Example17:** Check if a file exists or not. If its exists inform user it exists otherwise create it.

**[ -f /var/abc.txt ] && echo “file exists” || touch /var/abc.txt**

**Example18:** Check if my previous command executed successfully or not

**pwd**  
**/home/surendra**  
**[ $? -eq 0 ] && echo “Yes, pwd command successfully executed” || echo “pwd command failed”**

**! -NOT operator(Negation operator)**

Update:A follower(Flatcap) of this blog suggested below examples for this operator. This operator is very much handy when you want to delete or move all the files expect .txt files we can use this operator.

**Example18:** Remove all the files which are not a .doc file.

**ls | grep -v doc | xargs rm -rf**

or

**rm -rf \*.{xls, txt, pdf}**

The above rm command have some disadvantage, if we have more number of file extension we have to enter all of them. And the first command we are using total 4 commands to accomplish our task. so At this point we better use ! operator for removing all the files expect .doc files as shown below.

**rm !(\*.doc)**

**Example19:** Move all the files expect .doc files to /opt

**mv !(\*.doc) /opt/**

**Example20:** Copy all the files expect pdf files to /opt/all

**cp !(\*.pdf) /opt/all**

**| — PIPE operator**

This is well-known to many people, this operator is used to send output of first command as an input to second command.

**Example21:** Count no of files/folder located in a folder

**ls -l | wc -l**

**Example22:** Display all the partition names in the system.

**df -h | awk ‘{print $1}’**

**{ } –Command combination operator**

This operator used to combine two or more commands to be executed depending on the previous command. This can be explained with an example

Suppose I want to check if a file exists or not. If it does not exists We have to display error and then we have to create it. Lets do that without this {} operator

**Example23:** Check if /opt/abc.txt exists or not?

**[ -f /opt/abc.txt ] || echo “The file does not exists”; touch /opt/abc.txt**

touch command in the above command never depends on first command exit status. In order to make it depend on first command we have to use { } braces as shown below.

**[ -f /opt/abc.txt ] || { echo “The file does not exists”; touch /opt/abc.txt; }**

**Note:** Make sure that you given ; operator at the end of last command as shown in above example. This is mandatory and without it you may get error.

**Example24:** Ping to a machine, if able to ping display some useful info then try to connect to it using ssh.

**ping -c**[**www.linuxnix.com**](http://www.linuxnix.com/)**&& {echo “That is pingable, too good..!”; ssh root@**[**www.linuxnix.com;**](about:blank)**}**

**() –Precedence operator**

This operator used to execute command in precedence order which command to execute first. For example see below example

**command1 && command2 || command3 && command4**

If you see command2 executes if the command1 executed successfully. But once command2 executed remaining commands command3 and command4 are not going to execute. In order to eliminate this we have to use precedence operator. So below command sequence command3 and command4 will be executed if command1 or command2 failed.

**Example25:**

**( caommand1 && command2 ) || ( command3 && command4 )**

In this way we can save some valuable time. Please comment your thoughts on this.

**Some good stuff of one liner’s you can find it in below blogs and documents:**

**Sed One liners:**

<http://www-rohan.sdsu.edu/doc/sed.html>

**General one liner:**

<http://www.slac.stanford.edu/grp/cd/soft/unix/ug/unixtips.html>

<http://unix.stackexchange.com/questions/6/what-are-your-favorite-command-line-features-or-tricks>

<http://javarevisited.blogspot.com/2011/03/10-find-command-in-unix-examples-basic.html#ixzz2OQTuAX20>

**Find related examples:**

<http://javarevisited.blogspot.in/2011/03/10-find-command-in-unix-examples-basic.html>

find . -mtime 1  (find all the files modified exact 1 day)  
  
find . -mtime -1 (find all the files modified less than 1 day)  
  
find . -mtime +1 (find all the files modified more than 1 day)