

# Lezione 10 – Esercizi riassuntivi

## ESERCIZI

### Linux Introduction Puzzle/Quiz/Treasure Hunt

Taken from:

<http://its2.unc.edu/divisions/rc/training/scientific/>

Get the file *ncfmp.tgz* and untar and unzip it in your directory space.

```
https://homes.di.unimi.it/genovese/esercizi\_linux/ncfmp.tgz
cd Downloads
tar -xf ncfmp.tgz
```

Under the top *ncfmp* directory, create a directory called *answers*. You will be copying files to this directory throughout this exercise.

```
mkdir answers
```

For all the Linux commands discussed in this class there is a very useful command you can use to get extensive help on using any given command. We have emphasized using this command during this lecture. What is the name of this command? Find a directory in the *ncfmp* directory with the same name as this command. Go to that directory and copy the file you find there to your *ncfmp/answers* directory.

```
cd man
cp shaw ../answers
```

What file(s) do you find in the *ncfmp/HBL/runs* directory?

```
cd HBL/runs
ls
```

Which one has been modified most recently? Go into that directory.

```
ls -t
cd 1996_Fran_long
```

One of the files there is \*not\* a directory. Copy that file to your *ncfmp/answers* directory.

```
cp wrf ../../answers
```

What is the biggest file in the *ncfmp/dem* directory? Copy just the last line of that file to your answers directory into a file you create called *quiz.ans*. (Hint: you can do this on a single command line using the Linux command tail and a redirect).

```
cd ../../dem
ls --sort=size -l
tail -n 1 bathy_topo.out > ../../answers/quiz.ans
```

How do you list files such that file types are distinguished by the font color? (Hint: this is an option to a particular command).

```
ls --color=always
```

Find the file under the *ncfmp* directory which has the same name as the option above. Move this file to your answers directory.

```
cd ..  
find . -name "color"  
mv ./ww3/bin/color answers
```

Go to the *ncfmp/ww3/ftn* directory. List all the files that end with the *.f* extension. (Hint: see using wild cards). Now copy (just) these files to the answers directory.

```
cd ww3/ftn  
ls *.f  
cp *.f ../../answers
```

Go to the *ncfmp/ww3* directory. To see all the files under this directory you could issue the command:

```
find . -name "**"
```

If you wanted to count the files, you could pipe this output into the word count program, *wc*, like so:

```
find . -name "**" | wc -l
```

How many files do you count? Now remove the test directory and all of its contents (you can do this with one command). How many files are left?

```
rm -rf test  
find . -name "**" | wc -l
```

Find a file with this number under the top *ncfmp* directory and copy it to your answers directory. (Note: the file name is spelled out, for example 7 would be seven).

```
cd ..  
find . -name "two*"  
cp `find . -name "two*"` answers/  
(alt gr + apice)
```

Finally, to put all this together and to see the answer to this puzzle, go to your answers directory. You should have 12 files there. To arrange these in the proper order and to get a nicely formatted output we will use a little Linux magic (in the tradition of Arthur C. Clarke, magic is defined as something more advanced than the level of this course :). Issue the following command and read the output:

```
cat * | sort -n | tr -d '\n' | tr -d '[[:digit:]]' | tr '%' '\n' | tr -s ''
```

This simply concatenates all the files together (cat) then sorts (sort) them using a numeric key that I put in the text, and then does a series of character translations (tr) to make the output pretty by 1) removing new lines between the files, 2) removing the numbers used for sorting, 3) inserting new lines wherever I put a '%' character and finally 4) squeezing any extra spaces down to a single space.

### Linux scavenger hunt

Taken from:

<https://github.com/pushingice/scavenger-hunt>

#### Possibile problema con Python a causa della nuova versione di Ubuntu

Il comando python potrebbe restituire un errore perché di default la nuova versione di Ubuntu richiede python 3.

#### SOLUZIONE

\$ sudo apt install python2

\$ sudo update-alternatives --install /usr/bin/python python /usr/bin/python2 1

\$ sudo update-alternatives --install /usr/bin/python python /usr/bin/python3 2

\$ sudo update-alternatives --config python

(selezionare 1)

#### Esercizio

If you are using a new Linux install or Live CD, you may need to install Git first (*sudo apt-get install git* on Ubuntu). Open a terminal and type:

```
git clone https://github.com/pushingice/scavenger-hunt.git  
cd scavenger-hunt
```

First, choose a secret number with at least 4 digits to share with your team, or keep to yourself if you are working alone. Don't forget it! The secret number makes your clues unique, so other teams can't look over your shoulder. Then type:

```
python generate_clues.py [secret number]
```

Any time we enclose something in square brackets, you need to replace it with an actual value (called an argument). For example, to get started I might type:

```
python generate_clues.py 1234
```

This will create a subdirectory called clues. Be sure to keep this file (called the README) open in a separate viewer.

#### Clue 1: The Hunt Begins

*man*

The first command we are going to learn is *man*, which is short for manual. Typing *man [command]* will give you a help page (usually called a manpage) for most commands.

*ls*

The next command we need to learn is *ls* (list). Type *man ls* and read the description. Press *q* to exit. Then type *ls* and you should see something like this:

```
APPENDIX.md clues generate_clues.py LICENSE.md next_clue.py README.md
```

Items which are blue are directories and everything else is a file. Any time you need to know which files and directories are available, type *ls*.

*cd*

We need a couple more tools before we can start clue hunting. To change to another directory we use *cd* (change directory). You may notice that man *cd* doesn't work. Sometimes there is no manpage for a command. In that case google is your friend. Changing directories is pretty simple:

`cd clues`

This puts us in the *clues* directory. To go up a directory, we can do this:

`cd ..`

If you ever get lost, just do

`cd ~/scavenger-hunt`

to return home. If you *cd* to the *clues* directory and do an *ls*, you will notice that there are a lot of clue directories. Most of them contain fake clues. Throughout our hunt we will be looking for real clues. Using *cd*, navigate to *clues/12345* and type *ls*. You should see a single file named *clue*.

*cat*

Finally we need to be able to look at our clues. First read the manpage for *cat*, then do:

`cat clue`

This should list the clue in your terminal. From now on, everything we need will be contained in these clue files. It's a good idea to keep track of all the clue folders (like *12345*) on a piece of paper. You can also do things like copy all the clue files to your home folder, or cut and paste the clue text into another file.

`echo 12345 >> clues_collected.txt`

### Clue 2: The Lay of the Land

*pwd*

What if we get lost and need to know where we are? Just type *pwd* (print working directory). This should print something like this:

`/home/user/scavenger-hunt/clues/12345`

We are five folders deep, in a folder named *12345*.

*cd*

Change directory is extremely useful, but it can also be confusing. We already saw how you can move up one directory (*cd ..*) or down one directory (*cd [dir]*). You can move up or down any number of directories in a single command like this (won't actually work here):

```
cd ../../one/two/
```

You would navigate up 3 directories relative to where you are, then down into directory one and then two. This is what's known as a relative path: it depends on where you start where you will end up. The other way to change directories is with absolute paths. Try this:

```
cd /
```

Look around and see what's there. This is known as the root path. You can explore the entire file system from here, using just *ls* and *cd*.

To find the next clue, go to the */usr* directory and count the number of subdirectories. This is a hint to your next clue location.

```
cd /usr  
ls | wc
```

Go to the *scavenger-hunt* directory, and type

```
python next_clue.py [secret number] [next clue number] [hint]
```

So, if there were 9 directories, we would type

```
python next_clue.py 1234 3 12
```

since our secret number is 1234, we want to find clue 3, and our hint is 5. The location of our next clue should be printed.

```
44074  
echo 44074 >> clues_collected.txt
```

If you get the hint wrong, an incorrect clue will be printed. This is an example of what is known as GIGO (garbage-in, garbage-out). Helpful tip: you can open more than one terminal window, or more than one tab in your current window with *CTRL-SHIFT-T*.

*less*

Less is a program that allows you to view files in a terminal. Unlike *cat* you can scroll through the file using up, down, page up, \*etc\*. After you navigate to clue 3, do *less clue*. The name Less is a play on More, a similar program. More is older and is so named because you could press enter to see more text.

```
cd clues/44074  
less clue
```

### Clue 3: Humans vs. Machines

*Binary vs. Text*

There are two basic types of files: binary and text. Text can be read by both humans and the computer, and is sometimes referred to as "human-readable". For example, the file you are reading right now is text.

Binary is a number system that uses only 0 and 1 as digits. For example, 42 is represented as 101010 in binary. Each digit is called a "bit". Eight bits is called a "byte". There are 256 possible bytes ( $2^8$ ). Bytes are a fundamental unit of measurement in computing (\*e.g.\* file sizes are in bytes). Computers use a shorthand for each byte called "hexadecimal" or more briefly "hex". In hex there are sixteen digits, the usual 0-9 and also A-F. A is equal to 10, B to 11, \*etc\*. Sometimes we write a `0x` in front of a hex number to indicate we are using hex: 42 is `0x2A`.

If you ever look at a file and see a bunch of "garbage", you are probably looking at a binary file. The content isn't intended for you: it's for the computer. Binary files are sometimes referred to as "machine-readable".

### /bin

One place you can always find binaries on a Linux system is in `/bin`. These binaries are programs: if you `ls` in `/bin` you may recognize some of them (including `ls` itself). This is also a convenient way to get a list of commands.

If you want to see the garbage view of a binary, you can `cat` or `less` one of them. You can even `cat cat` or `less less`. On some Linux systems you can see the hex itself with `hexdump`.

### /etc

This directory is named after the latin \*et cetera\* but is usually pronounced "et see". There are many text (and some binary) files here that are used to configure the system. Humans and computers can both read these files to find out how to configure the system.

For example, look at the `/etc/fstab` file. This describes how the filesystem is mapped to the hard drive.

Your hint for clue 4 is the file `/etc/hostname`. This file contains a single word, which is the name of your computer. This name is your hint. Remember we can find the next hint by typing:

```
python next_clue.py [secret number] [next clue number] [hint]
(es.) python next_clue.py 1234 4 ruggero-VirtualBox
00750
echo 00750 >> clues_collected.txt
cd clues/00750
cat clue
```

### Clue 4: Moving Day

#### *Making Space*

We've been exploring the directories that already exist on the computer. But what if we want to make our own folders and files? The first thing we need to do is create a new directory. First go home: `cd ~/scavenger-hunt`. Then do

```
mkdir saved-clues
```

What we're going to do is save off all the clues we find in a separate folder that we created with `mkdir` (make directory). Since the README is clue 1 we don't need to worry about it. If you've been writing down all the clue locations, this next part should be easy.

### *Stop Copying Me*

Let's copy all of the clues we've found so far to our saved-clues folder:

```
cp clues/44074/clue saved-clues/clue2  
cp clues/00750/clue saved-clues/clue3
```

This copies (*cp*) each clue to the new folder and gives them new names. If we had just done this

```
cp clues/44074/clue saved-clues/  
cp clues/00750/clue saved-clues/
```

The second file would overwrite the first, because they have the same name.

### *Keep Your Options Open*

Linux commands often have options that change how they behave. For instance, compare *ls -l* to ordinary *ls*. Here the *-l* is an option. You can group options together like this:

```
ls -lahS
```

The best way to find out about options is the manpage.

### *Moving On*

Now let's say we don't like the folder name *saved-clues*. We can just move (*mv*) it:

```
mv saved-clues [pick a new name]
```

Now do an *ls* to see the results of the move. Be careful with *mv*: you can easily overwrite an existing folder.

**Read the man page for *mv* and find an option to prevent overwriting. That option is your next hint.**

```
python next_clue.py 1234 5 n  
91098  
echo 91098 >> clues_collected.txt  
cd clues/91098  
cat clue
```

### Clue 5: Is There an Echo in Here?

#### *echo*

Sometimes we want the computer to repeat back the results of some command. Try:

```
echo hello
```

The most basic thing *echo* will do is repeat back whatever you type.

#### *Redirect*

You can use this to create a small file, or start a new file:

```
echo My bologna has a first name > oscar.txt
```

If you look in *oscar.txt* you will see exactly what you typed. The `>` symbol used here is called a redirect. It redirects whatever would normally be printed to the screen to a file. You can try it with other commands:

```
ls > my_directory.txt
```

You can also use *echo* to display what are called environment variables

```
echo $PATH  
echo $HOME
```

The *HOME* variable should make sense at this point.

The *PATH* variable tells the computer where programs are. Each path that could contain a program is placed between colons. Your hint for the next clue is the first path listed in your *PATH*.

```
echo $PATH  
python next_clue.py 1234 6 /usr/loca/sbin  
93927  
echo 93927 >> clues_collected.txt  
cd clues/73648  
cat clue
```

### Clue 6: Which `which` is which?

*which*

In the previous clue we learned about *PATH*. This tells the computer where to find binary programs we can run. But, how do we find where a specific program is? The answer is *which*:

```
which mv
```

This should print */bin/mv*. This tells us that the *mv* command is installed in the */bin/* directory. *which* itself is a program so you could try:

```
which which
```

*python* is a program we have been using to generate and test our clues. Your next hint is the location of the *python* program.

```
which python  
python next_clue.py 42 7 /usr/bin/python  
67670  
echo 67670 >> clues_collected.txt  
cd clues/67670  
cat clue
```

### Clue 7: Make Me a Sandwhich

<https://xkcd.com/149/>

*sudo*

Linux has the concept of a *root* user, which is similar to the administrator user on Windows. This user is sometimes called the super user. If you want to do something as the super user, but stay logged in as yourself, there is a command for that: *sudo*. It stands for "super-user do".

### *Installing Software*

Sometimes you need a new program. To install software on some versions of Linux (Ubuntu and Debian), you use the command *apt-get*. On other versions (Fedora, CentOS) you use the command *yum*. Let's install a text editing program called *vim*.

```
apt-get install vim
```

You should get an error message asking if you are root. This means you don't have the ability to install software. Instead, try:

```
sudo apt-get install vim
```

Now we have the ability to edit files. Try:

```
vim README.md
```

from the *scavenger-hunt* directory. Some of the commands for *vim* are a little strange. For now, just type *:q!* to quit.

The hint is the name of the first folder listed in */sys/kernel/debug*.

```
sudo -s
cd /sys/kernel/debug
ls (ritorna acpi)
exit
python next_clue.py 1234 8 acpi
67157
echo 67157 >> clues_collected.txt
cd clues/67157
cat clue
```

*Help. I can't sudo*

Depending on the system you are using, you may not have permission to use *sudo*. In this case you can use the hint *denied*.

### Clue 8: Counting Words

*wc*

Word count (*wc*) is a useful program. You can use it to tell how many lines, words, and/or characters are in a file:

wc README.md

This will print the number of lines, words, and characters, in that order. If you just want one of those, you can use *-l*, *-w*, or *-c*.

Check to see if you have the file */usr/share/dict/words* installed. If not, run this:

```
cat /usr/share/dict/words  
sudo apt-get install ispell
```

Now there is a file that acts as a dictionary for spell-checking: */usr/share/dict/words*. Your next clue is the number of words in the dictionary.

```
wc /usr/share/dict/words  
102774  
python next_clue.py 42 9 102774  
08394  
echo 08394>> clues_collected.txt  
cd clues/08394  
cat clue
```

### Clue 9: Searching High and Low

*grep*

Searching files is another useful trick. Try this:

```
grep secret README.md
```

This will print out every line that contains the word "secret". *grep* stands for Gnu Regular Expression Parsing. Gnu is an umbrella organization that publishes open source software. A regular expression is a pattern that matches text. In this case our regular expression is just "secret", and will only find exact matches. Regular expressions can be more powerful and/or complicated. For example:

```
grep m.n README.md
```

will find any line where the letters *m* and *n* exist with a single character between them. Check the man page for many interesting features of *grep*.

The next hint is the word that occurs after "tactful" in */usr/share/dict/words*. There's a specific option for *grep* that will make this easy.

```
grep -A 1 tactful /usr/share/dict/words  
tactfully  
python next_clue.py 1234 10 tactfully  
76649  
echo 76649>> clues_collected.txt  
cd clues/76649  
cat clue
```

### Clue 10: Pipes

#### *Piping Information*

Many commands will print their output. This is called "standard output" or *stdout*. We saw earlier how you can redirect standard output to a file ('>'). There is also standard input ('<'). For example, *cat < README.md* is the same as *cat README.md*. But standard input and output can be chained together using pipe ('|'). For example, you can count the number of files and folders in a directory like this:

```
ls | wc -w
```

This works by taking the output of *ls* and using it as the input to *wc*. Another example:

```
grep ^sand /usr/share/dict/words | wc -l
```

will print the number of words that start with "sand". The carat '^' symbol is a regular expression that means "starts with". You can also use '\$' for "ends with".

#### *Sort*

Sometimes you need to sort data alphabetically. You'll notice that the dictionary file is already sorted. You can create your own unsorted copy like this:

```
sort -R /usr/share/dict/words > random_words
```

Now you can sort *random\_words* to get back to alphabetical order, or *sort -r random\_words* for reverse alphabetical order.

Use the command *ls -la /usr* to get a big list of files. The 5th column in that list is the size of the file in bytes. Find the *sort* command to print the list of files with the largest file first, and then the rest in descending order. Your hint is the options you had to use. You'll need to use double quotes for your hint. For example, if your command was 'sort -a -b -c', your hint would be:

```
python next_clue.py [secret] 11 "-a -b -c"
```

```
ls -la /usr | sort -k 5 -n -r
python next_clue.py 1234 11 "-k 5 -n -r"
02980
echo 02980 >> clues_collected.txt
cd clues/02980
cat clue
```

### Clue 11: The Final Frontier

#### *Finding the Final Clue*

Using everything you've learned so far, and the fact that the real clues are different from the fake clues, find the final clue!

12345  
02502  
27503

22322  
73648  
67670  
89218  
08694  
42193  
02980

```
cp clues/44074/clue saved-clues/clue01
cp clues/00750/clue saved-clues/clue02
cp clues/91098/clue saved-clues/clue03
cp clues/93927/clue saved-clues/clue04
cp clues/58223/clue saved-clues/clue05
cp clues/67157/clue saved-clues/clue06
cp clues/08394/clue saved-clues/clue07
cp clues/08694/clue saved-clues/clue08
cp clues/42193/clue saved-clues/clue09
cp clues/02980/clue saved-clues/clue10
```

(Non so se è la soluzione pulita che aveva in mente l'autore...)

```
cd ~/scavenger-hunt
grep -Ril "clue 12" ./clues
21864
echo 21864 >> clues_collected.txt
cd clues/21864
cat clue
```

### Clue 12: Success!

*You Found All the Clues*

This is just the start of learning Linux. Explore, google things, break things. You can also start learning shell scripting (executing multiple commands), or even Python programming. Just type:

```
python
```

and you'll get an interactive programming environment. You can use *vim* or a text editor to write your python code, then execute it:

```
python my_code.py
```