%title: Linux Dojo - Linux terminal Hot Tips %author: Michele Morelli %date: 15th March 2020

-> # Introduction <-

-> # The command line is full of surprises <-

- More things than one may think can be done efficiently (and for free) from the command line:
  - PDF editing (e.g. pdftk);
  - image editing (e.g. imagemagick)
  - browsing the Internet (e.g. lynx)
  - Word processing (e.g. vim + LaTEX)
  - testing RESTful API (e.g. curl)
  - *writing presentations!* :-) (e.g. mdp)

-> # The 'Unix Philosophy' <-

This is made possible by the "*Unix philosophy*", initially defined by Ken Thompson: many little, robust programs that do one thing well.

-> Excerpt from The Art of Unix Programming: <-

- **Make each program do one thing well.** To do a new job, build afresh rather than complicate old programs by adding new "features".
- **Expect the output of every program to become the input to another, as yet unknown, program.** Don't clutter output with extraneous information. Avoid stringently columnar or binary input formats. Don't insist on interactive input.

-> # More philosophy <-

-> **There is more than one way to do it ** TIMTOWTDI (Tim Toady)** <-

Originally this was the motto of the Perl community, but it applies well to general Linux scripting in my opinion.

In contrast with Python's Zen: "There should be one â€” and preferably only one â€” obvious way to do it."

-> # Linux pipelines and functional programming <-

Linux pipelines and functional programming languages' pipelines are very similar! We start from an 'immutable' value (usually a string) and we pass it through a series of functions, which take a string-argument and return a string value (in most cases).

```
$ Data source => S | S => S | S => S

$ Data source => S | S => S | S => S >> file.txt

$ $(Data source => S | S => S | S => S >> file.txt)
```

Some important caveats:

- no guarantee of referential transparency;
- the values are not really immutable!

-> # Linux pipelines and functional programming <-

Some typical string generators:

- $cat (to read from files)
- $echo (to pass strings)

... but really, the sky is the limit!

-> # Example 1: <- Our manager asked us to make a short report of all the commits that were done for the code-dojo

repository.

Our manager told us that the report must follow some *specific format requirements*:

- it should contain only the commits' hash codes;
- each hash code should contain only the first 6 characters;
- the output should be all-caps;

We are also told that we need to create a command that sends us back to the first commit in the repository.

How can we do this with a one-liner?

EXPECTED OUTPUT EXAMPLE:

adas90 34KSDL DSSA87

-> # curl and REST API testing <-

- curl is a very powerful tool that allows to make requests using different protocols (e.g. FTP, HTTP...) and methods (GET, POST, PATCH)
- it is invaluable when testing RESTful APIs!

-> # Example 2: <-

We need to test an endpoint of a new REST API that we are evaluating, and we want to ensure that the requests using the GET method return meaningful results.

We need to check that the following endpoint: **https://jsonplaceholder.typicode.com/todos/**

returns the correct information for items.

For example, to get information about the item with ID one, we would make a get call to:
https://jsonplaceholder.typicode.com/todos/1

How can we make get the response codes for items *with IDs from 35 to 47 only*?