The Linux Console

linuxcommand.org

Now what?

You have Linux installed and running. The GUI is working fine, but you are getting tired of changing your desktop themes. You keep seeing this “terminal” thing.

The Console is your friend

Let's type in something to get started

```
damiano@damiano-Z97P-D3:~$ welcome
welcome: command not found
```

nice!

You're not logged in as root, are you?

If the last character of your shell prompt is # rather than $, you are operating as the superuser. This means that you have administrative privileges

What is damiano@damiano-Z97P-D3:~$?

- damiano is the current user name
- damiano-Z97P-D3 is the host
- ~ is the current directory. In this case, the home directory, /home/damiano

Hello, World!

To invoke a program, just type its name

```
damiano@damiano-Z97P-D3:~$ echo hello world
hello world
```

`echo` is a binary **executable**: it resides in `/usr/bin`. `hello` and `world` are **arguments**

Moving around

We use those 3 commands to navigate the file-system. Make sure you remember them:

- `ls` lists directory contents
- `pwd` prints the name of the current/working directory
- `cd` changes directory

pwd

The directory you are standing in is called the **working directory**

```
damiano@damiano-Z97P-D3:~$ pwd
/home/damiano
```

ls

To list the files in the working directory, use the `ls` command.

```
damiano@damiano-Z97P-D3:~$ ls
file1
file2
file3
```

Change directory

`cd` changes your working directory.

```
damiano@damiano-Z97P-D3:~$ cd /usr/bin/
damiano@damiano-Z97P-D3:/usr/bin$ pwd
/usr/bin
damiano@damiano-Z97P-D3:/usr/bin$ ls
[
2to3
2to3-2.7
2to3-3.4
7z
```

Absolute path

O.K., now let's say that we wanted to change the working directory to the parent of `/usr/bin` which is `/usr`. With an absolute pathname:

```
damiano@damiano-Z97P-D3:/usr/bin$ cd /usr
damiano@damiano-Z97P-D3:/usr$ pwd
/usr
```

Relative path

Here is illustrated a relative pathname:

```
damiano@damiano-Z97P-D3:~$ cd /usr/bin/
damiano@damiano-Z97P-D3:/usr/bin$ cd ..
damiano@damiano-Z97P-D3:/usr$ pwd
/usr
```

Some more complex relative paths

```
damiano@damiano-Z97P-D3:~$ cd /usr/bin
damiano@damiano-Z97P-D3:/usr/bin$ cd ..
damiano@damiano-Z97P-D3:/usr$ cd ./
damiano@damiano-Z97P-D3:/usr$ cd ./bin/
```

Useful shortcut

If you type `cd` followed by nothing, `cd` will change the working directory to your home directory

Looking Around

List the files in the working directory

- `ls`

List the files in the /bin directory

- `ls /bin`

List the files in the working directory in long format

- `ls -l`

List the files in the /bin directory and the /etc directory in long format

- `ls -l /etc /bin`

List all files in the parent of the working directory in long format

- `ls -la ..`

less

`less` is a program that lets you view text files.

This is very handy since many of the files used to control and configure Linux are human readable.

Press `Q` to exit

file

```
file /bin/echo --mime-type
```

`file` is a program for inspecting files, `/bin/echo` is the file we want to know something about, `--mime-type` is a flag

Touching files

`touch` creates empty files

```
damiano@damiano-Z97P-D3:~$ touch new\ empty\ file
damiano@damiano-Z97P-D3:~$ file new\ empty\ file
new empty file: empty
```

We use \ to escape white spaces.

Getting help

Command line programs ships with a self contained manual. It comes handy very often, just try:

```
file --help
```

There is also a program for reading manuals: it's called `man`

```
man echo
```

You can even try `man man`. Press `Q` to exit.

A Guided Tour

- The root directory / is where the file system begins. The root directory only contains subdirectories.
- /boot is where the Linux kernel and boot loader files are kept. The kernel is a file called vmlinuz.
- The /etc directory contains the configuration files for the system. All of the files in /etc should be text files.

- `/bin` and `/usr/bin` contain most of the programs for the system.
- The `/sbin`, `/usr/sbin` directories contain programs for system administration, mostly for use by the superuser.
- The `/usr` directory contains a variety of things that support user applications.

- `/usr/local` and its subdirectories are used for the installation of software and other files for use on the local machine.
- The `/var` directory contains files that change as the system is running. This includes `/var/log`.
- The shared libraries (similar to DLLs in that other operating system) are kept in `/lib`.

- `/home` is where users keep their personal work.
- `/root` is the superuser's home directory.
- `/tmp` is a directory in which programs can write their temporary files.

- The `/dev` directory is a special directory, since it does not really contain files in the usual sense. Rather, it contains devices.
- The `/proc` directory is also special. This directory does not contain files. In fact, this directory does not really exist at all. It is entirely virtual
- Finally, we come to `/media`, a normal directory which is used for mount points

Manipulating Files

Some commands:

- `cp` copy files and directories
- `mv` move or rename files and directories
- `rm`remove files and directories
- `mkdir` create directories

Wildcards

- `*` Matches any characters
- `?` Matches any single character
- `[characters]` Matches any character that is a member of the set characters. The set of characters may also be expressed as a POSIX character class

Wildcards for filenames

All filenames

- `*`

All filenames that begin with the character “g”

- g*

All filenames that begin with the character “b” and end with the characters “.txt”

- `b*.txt`

cp

The cp program copies files and directories. In its simplest form, it copies a single file:

```
cp file1 file2
```

It can also be used to copy multiple files (and/or directories) to a different directory:

```
cp file1 file2 directory
```

mv

The `mv` command moves or renames files and directories depending on how it is used

```
mv filename1 filename2
```

To move files (and/or directories) to a different directory:

```
mv file... directory
```

Be careful with rm!

Linux does not have an undelete command. Once you delete something with `rm`, it's gone.

mkdir

The `mkdir` command is used to create directories. To use it, you simply type:

```
mkdir directory...
```

Folders are files?

Commands With Wildcards

```
cp *.txt text_files
```

Copy all files in the current working directory with names ending with the characters ".txt" to an existing directory named `text_files`.

```
mv my_dir ../*.bak my_new_dir
```

Move the subdirectory `my_dir` and all the files ending in ".bak" in the current working directory's parent directory to an existing directory named `my_new_dir`.

```
rm *~
```

Delete all files in the current working directory that end with the character “~”.

Please don’t delete all your files in the home

```
rm -rf /
```

Ooops! We don't have enough power to do that!

I/O Redirection

A powerful feature used by many command line programs called *input/output redirection*

many commands such as ls print their output on the display

we can redirect the output of many commands to files, devices, and even to the input of other commands

Standard Output

By default, standard output directs its contents to the display

```
ls > file_list.txt
```

the `ls` command is executed and the results are written in a file named `file_list.txt`

>>

If you want the new results to be appended to the file instead, use >> like this:

```
ls >> file_list.txt
```

Standard Input

By default, standard input gets its contents from the keyboard, but like standard output, it can be redirected

```
sort < file_list.txt
```

The results are output on the display since the standard output was not redirected. We could redirect standard output to another file like this:

```
sort < file_list.txt > sorted_file_list.txt
```

Now I'm really confused!

Pipelines

With pipelines, the standard output of one command is fed into the standard input of another

```
ls -l | less
```

grep

Examines each line of data it receives from standard input and outputs every line that contains a specified pattern of characters.

Original document:
http://linuxcommand.org/lc3_learning_the_shell.php