ICTP DP Linux Basic Course

ESP Students - First Semester

Graziano Giuliani
ggiulian@ictp.it

The Abdus Salam International Centre for Theoretical Physics

ICTP Diploma Program
August 8, 2022

Course Outline [1]

Daily program

- **UNIX/Linux**
  1. The Shell Command Line
  2. Basic command line programs
  3. Basic file editing
- Intermediate command line commands
- Editing and compiling source code
- Text file manipulation and shell scripting

Slides:

`http://tinyurl.com/2jsvfbd6`

or the LaTeX source on GitHub:

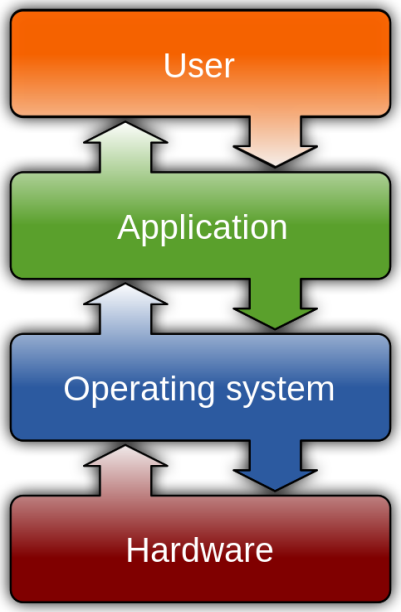
`https://github.com/graziano-giuliani/LinuxBasics`

[1] Course created in 2019 with Adriano Angelone, now LPTMC-FR

The origin of UNIX

Operating System

- Hardware
  - CPU, GPU, Motherboard, Cabling, Power
- User
  - Keyboard, Mouse, Monitor, etc
- Software
  - Application, Operating system



- 1969 - Multi-Tasking Multi-User Unix AT&T
- 1991 - Free Unix OS for Intel X86 (Linux)

By Peter Hamer: Ken Thompson (sitting) and Dennis Ritchie at PDP-11 Magnus Manske, CC BY-SA 2.0

The UNIX Philosophy

Why UNIX is such a good idea

1. Make each program do one thing well.
2. Expect the output of every program to become the input to another.
3. Purpose of computation is data transformation

. . . at its heart is the idea that the power of a system comes more from the relationships among programs than from the programs themselves. Many UNIX programs do quite trivial things in isolation, but, combined with other programs, become general and useful tools. . . .

The UNIX Programming Environment, Brian Kernighan and Rob Pike, 1984

The Linux Revolution

Why a free OS is a good idea

```
From: torvalds@klaava.Helsinki.FI (Linus Benedict Torvalds)
Newsgroups: comp.os.minix
Subject: What would you like to see most in minix?
Summary: small poll for my new operating system
Message-ID: <1991Aug25.205708.9541@klaava.Helsinki.FI>
Date: 25 Aug 91 20:57:08 GMT
Organization: University of Helsinki


Hello everybody out there using minix -

I'm doing a (free) operating system (just a hobby, won't be big and
professional like gnu) for 386(486) AT clones.  This has been brewing
since april, and is starting to get ready.  I'd like any feedback on
things people like/dislike in minix, as my OS resembles it somewhat
(same physical layout of the file-system (due to practical reasons)
among other things).

I've currently ported bash(1.08) and gcc(1.40), and things seem to work.
This implies that I'll get something practical within a few months, and
I'd like to know what features most people would want.  Any suggestions
are welcome, but I won't promise I'll implement them :-)

                Linus (torvalds@kruuna.helsinki.fi)

PS.  Yes - it's free of any minix code, and it has a multi-threaded fs.
It is NOT protable (uses 386 task switching etc), and it probably never
will support anything other than AT-harddisks, as that's all I have :-(.
```

The Free Software Movement

Free Software as in Freedom

The Freedom to

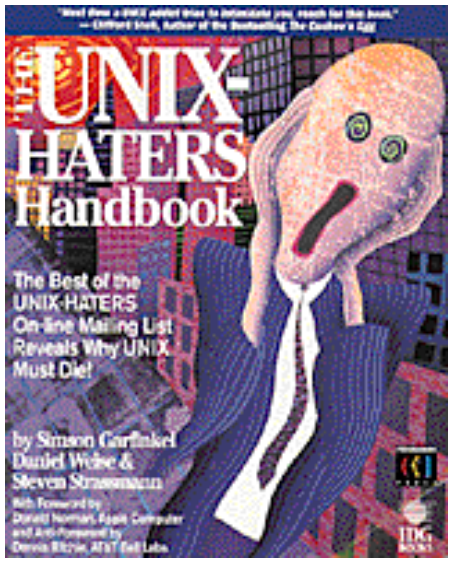
- run the program as you wish, for any purpose
- study how the program works, and change it as you wish
- redistribute copies so you can help others
- distribute copies of your modified versions to others

GPL v3
Free Software

Free as in Freedom

Linux architecture

Linux is the kernel and we do not hate it!

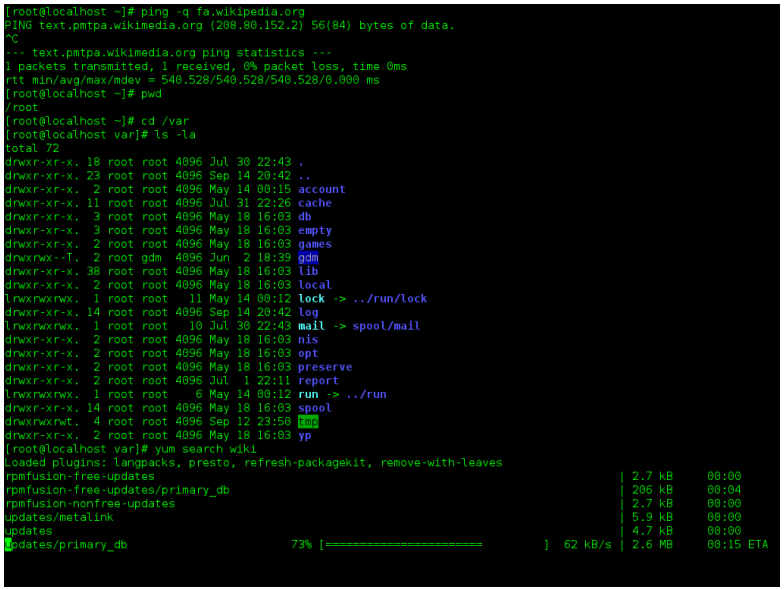


- The Linux kernel is a program that loads all the other programs and supervise the resources allocated to each one of them, providing locking and I/O services.
- The system services *(daemons)* are programs running on the system and providing facilities that allow or enhance access to system resources.
- User programs are controlled interactively or through batch job submission system by physical users concurrently accessing system resources through a multi tasking sharing of the CPU(s).

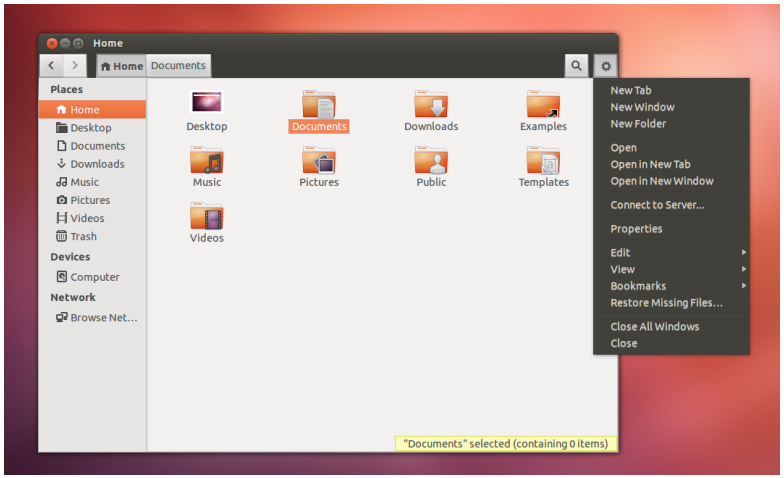
The Command Line

The old terminal

Command Line Interface (CLI): programmable, steep(-ish) learning curve



Graphical User Interfaces (GUI): intuitive, difficult to program



Authentication

Multi-User

To support multiple users, Linux require authentication before authorizing the user programs to allocate system resources.

- **Authentication**: The user is authenticated with username/password challenge by a login program.
- **Authorization**: The system creates an environment by providing the set of system resources the user may access
- **Allocation**: The user access the resources by running programs through a command interpreter.

The command shell

How it works

The SHELL is a text based command interpreter

- waits for the user command input showing up a prompt
- controls the user environment through variables
- executes user commands managing the input, output and error streams

There is not just a single shell program!

Linux Program

Running a program in the CLI

The User must type in a command line at the CLI prompt

A typical UNIX command line contains:

- The name of the program executable
- The options modifying the execution
  - short format: -f
  - long format: - - a_longer_string
- The argument or list of arguments

Options and arguments may have convenient defaults!

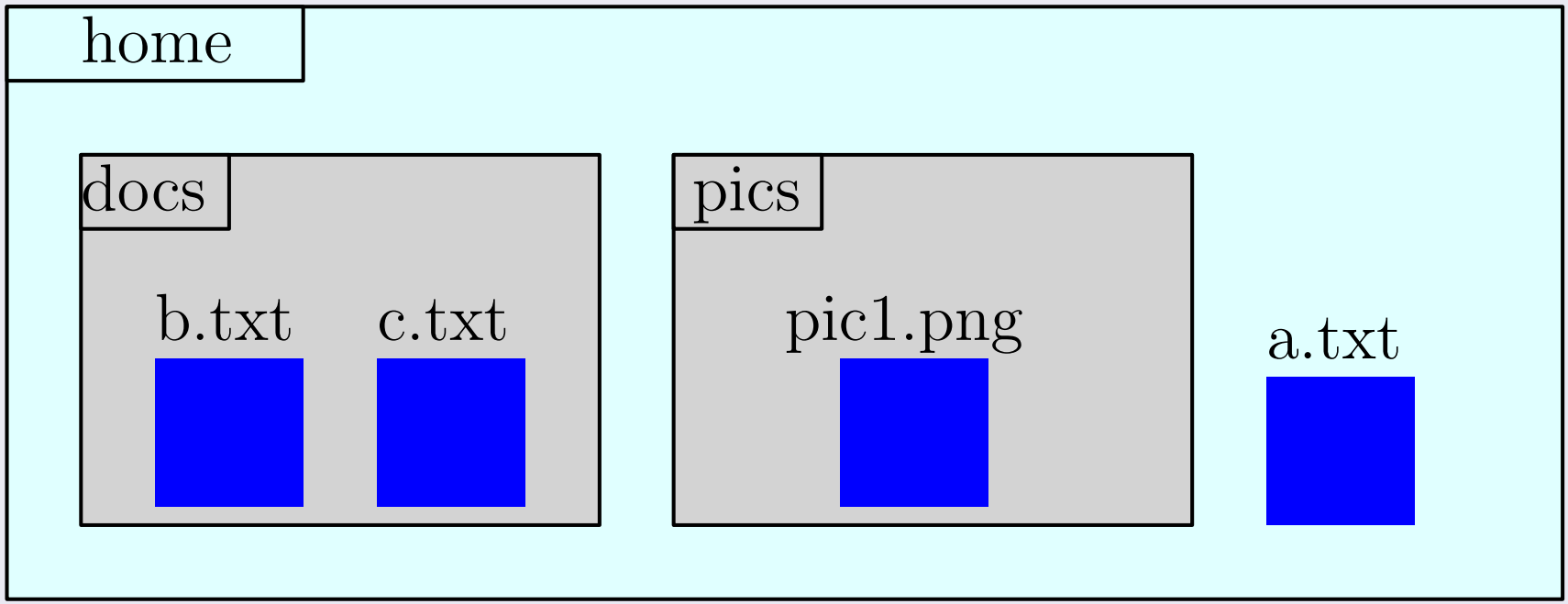
As an example, these are valid syntax for command lines:

```
ls -l Documents
```

```
cp --force theorem.tex Documents
```

Directories contain files, files contain information

`/` is the filesystem root directory



Directories in a path are separated by `/`

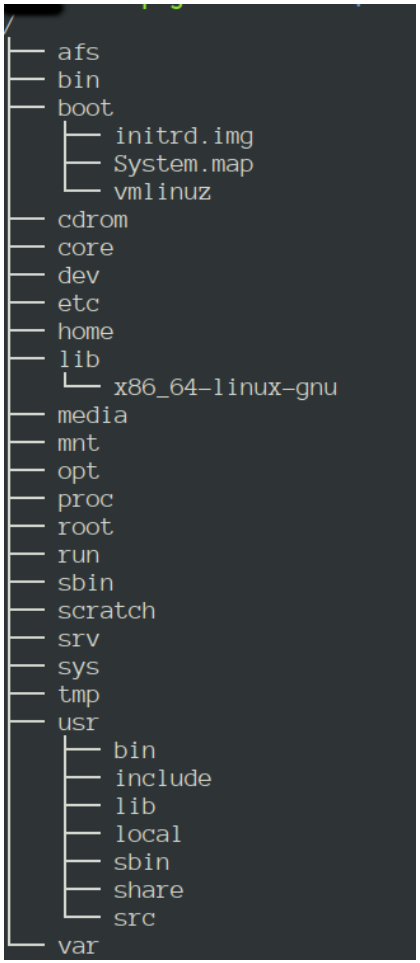
Files and directories have a **full path** in the filesystem:

`/home/docs/c.txt` FQP: Fully Qualified Path

The file system

Filesystem Hierarchy Standard (FHS)

All files and directories appear under the root directory `/`, even if they are stored on different physical or virtual devices.



- `bin` : binaries, contains system executables
- `lib` : libraries, contains shared or static pieces of code which are used by running executables or to create executables
- `tmp` : system or user temporary files
- `etc` : configuration files
- `home` : user files (one directory per user)

Home directory

`.` : the current directory
`..` : the parent of the current directory
`/home/{user}` is the default CWD

Commands

Pathfinding

Let us try to run the basic programs to navigate the Filesystem:

Print the current directory FQP

`pwd`

`pwd` prints the **fully qualified path** of the current directory

```
[aangelon@login02 ~]$ pwd
/home/aangelon
[aangelon@login02 ~]$
```

In commands, you use by default the **relative path** respect to the **CWD**:
`a.txt` without a path is a file in the current working directory

Have you noticed?

`pwd` does not require options or arguments

`CWD` stands for Current Working Directory

Commands

Moving around

List a file or the files in a directory

```
ls {directory or file}
```

```
[aangelon@login02 ~]$ ls
arch  devil  entham  example_file  intel  lpmc  scripts
[aangelon@login02 ~]$
```

Change the working directory

```
cd {directory}
```

```
[aangelon@login02 ~]$ ls
arch  devil  entham  example_file  intel  lpmc  scripts
[aangelon@login02 ~]$ cd intel
[aangelon@login02 intel]$ ls
ism
[aangelon@login02 intel]$
```

Have you noticed? Default arguments and options!

`ls` without arguments list the content of the **CWD**

`cd` without arguments change the CWD to the user home directory

Commands

Creating new filesystem objects

Create new directories

`mkdir <directories>`

```
[aangelon@login02 ~]$ ls
arch  devil  entham  example_file  intel  lpmc  scripts
[aangelon@login02 ~]$ mkdir new_dir
[aangelon@login02 ~]$ ls
arch  devil  entham  example_file  intel  lpmc  new_dir  scripts
[aangelon@login02 ~]$
```

Create new (empty) text files

`touch <filenames>`

```
[aangelon@login02 ~]$ ls
arch  devil  entham  example_file  intel  lpmc  new_dir  scripts
[aangelon@login02 ~]$ touch new_example_file
[aangelon@login02 ~]$ ls
arch  devil  entham  example_file  intel  lpmc  new_dir  new_example_file  scripts
[aangelon@login02 ~]$
```

Have you noticed?

A plural is specified above. Guess what it means?

Commands

Removing existing filesystem objects

removes files and directories

`rm <filenames>`

`rm -r <directories>`

```
[aangelon@login02 ~]$ ls
arch  devil  entham  example_file  intel  lpmc  new_dir  new_example_file  scripts
[aangelon@login02 ~]$ rm new_example_file
[aangelon@login02 ~]$ ls
arch  devil  entham  example_file  intel  lpmc  new_dir  scripts
[aangelon@login02 ~]$ rm -r new_dir
[aangelon@login02 ~]$ ls
arch  devil  entham  example_file  intel  lpmc  scripts
[aangelon@login02 ~]$
```

ATTENTION!

The remove operation cannot be undone. There is no Trash directory.
Be careful, especially in using the recursive `-r` option!!!

Manual

What are all the possible options?

Reading the manual pages

In CLI mode, you can access the manual page for each command in text format.

```
man <command>
```

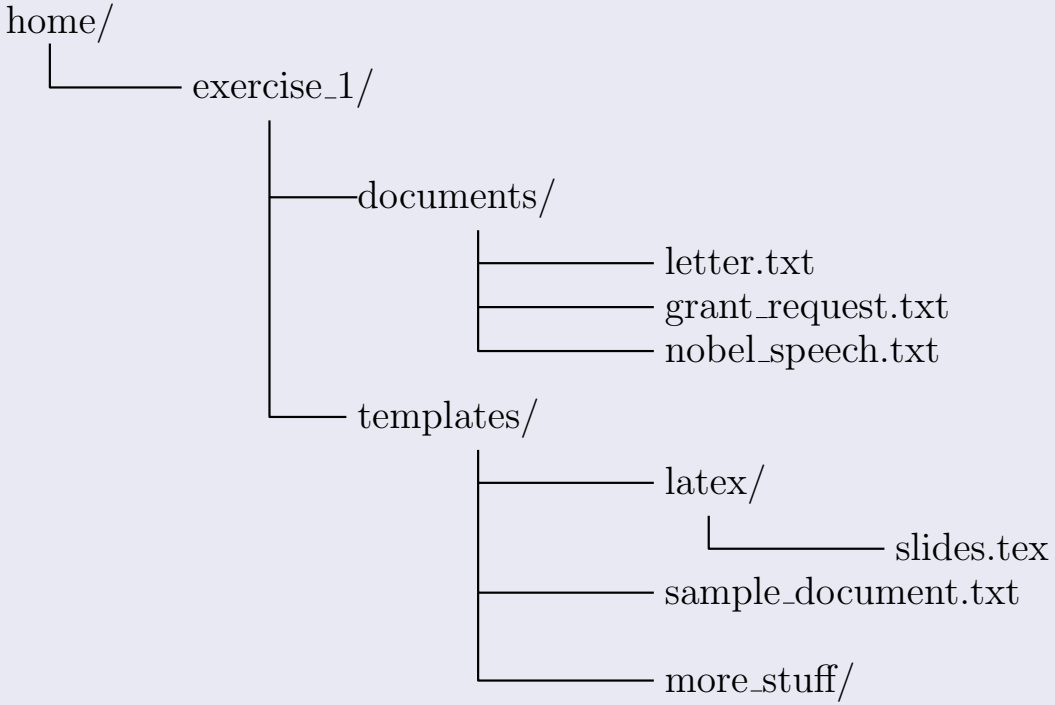
ATTENTION!

The formatting of the manual page on the text window is a complex operation and depends on the available lines/rows of the terminal. To be compatible with old terminals, the best window size ifor the man program is at **80rows x 24 lines**.

Exercise I

cd, mkdir, touch, rm

Using the commands we have seen, create the below directories and files, and then remove them all (**after carefully checking**):



How to check?

```
ictp-install tree
tree home
```

Solution to Exercise I

Creation part

```
cd
mkdir -p home/exercise_1/documents
mkdir -p home/exercise_1/templates/{latex,more_stuff}
cd home/exercise_1/documents
touch letter.txt grant_request.txt nobel_speech.txt
cd ../templates
touch latex/slides.tex sample_document.txt
cd
```

Removal part

```
cd
rm -r home
```

Commands

Moving and copying files

Copy an object to another location

`cp <old_path> <new_path>`

```
[aangelon@login02 ~]$ ls
arch  devil  entham  example_file  intel  lpmc  scripts
[aangelon@login02 ~]$ ls intel
ism
[aangelon@login02 ~]$ cp example_file intel/
[aangelon@login02 ~]$ ls intel
example_file  ism
[aangelon@login02 ~]$
```

```
[aangelon@login02 ~]$ ls
arch  devil  entham  example_file  intel  lpmc  scripts
[aangelon@login02 ~]$ cd intel/
[aangelon@login02 intel]$ ls
ism
[aangelon@login02 intel]$ cp ../example_file ./example_file_2
[aangelon@login02 intel]$ ls
example_file_2  ism
[aangelon@login02 intel]$
```

Attention! `<new_path>` is overwritten and lost!

Suggestion: use `cp -i`

`cp -r`: copy entire directories

Move an object to another location

`mv <old_path> <new_path>`

Same syntax as `cp`, `old_path` is removed after copy.

Commands

Finding and listing files

Recursively searches files in a directory

```
find <directory> <options>
```

- `-name`:
  specify file name (no paths here)
- `-path`:
  specify (part of) the file path
- `-printf %{format}`:
  print details of the items found
- `-delete`:
  deletes the files found

```
~/example_folder » ls
file_1  file_2  file_3  subfolder
-------------------------------------------------------
~/example_folder » ls subfolder
file_1  file_2
-------------------------------------------------------
~/example_folder » find . -name 'file_1'
./subfolder/file_1
./file_1
-------------------------------------------------------
~/example_folder » find . -path '*/file_1'
./subfolder/file_1
./file_1
-------------------------------------------------------
~/example_folder » find . -path '*/*/file_1'
./subfolder/file_1
```

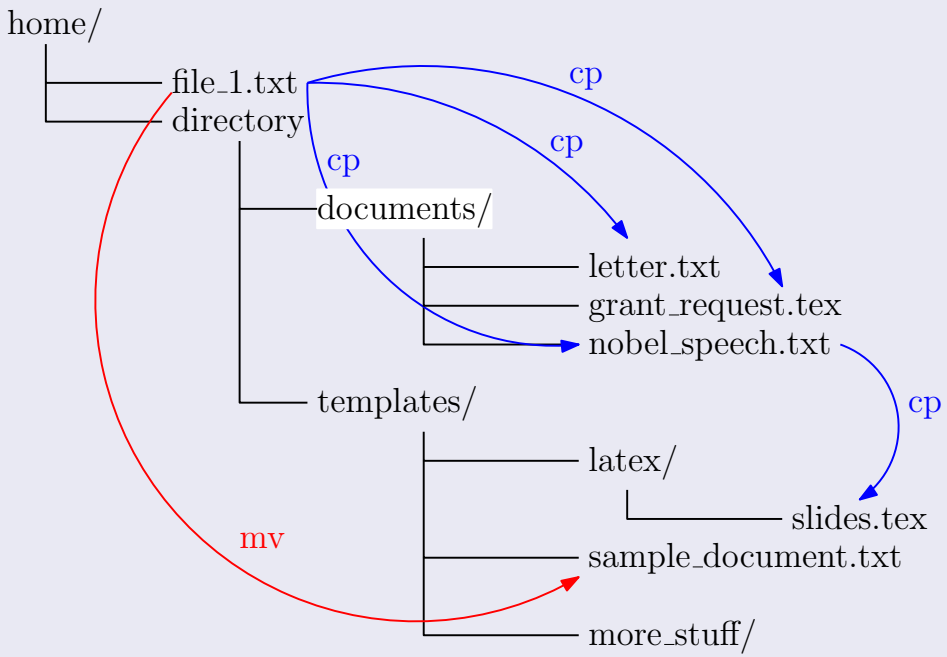
You can use wildcards

`*` can replace any character (more in the future)

Exercise II

cp, mv, find

Using the commands you know, create these directories and files, copying and moving files as shown:



Then, using `find`, show the location of all `.tex` files

Be careful with the order of the operations!