# Linux Basics I:
# What is Linux, the filesystem

Adriano Angelone, Graziano Giuliani

# Course Outline

- **UNIX/Linux Basics**
- Intermediate shell commands
- Editing and compiling source code
- Text file manipulation
- Basic shell scripting

Download slides and exercise files with the command

```
git clone https://github.com/AA24KK/LinuxBasics.git
```

or download a ZIP archive at

```
https://github.com/AA24KK/LinuxBasics/archive/master.zip
```

Adriano: **aangelon@ictp.it**, Room 263, ICTP
Graziano: **ggiulian@ictp.it**

# The Origin of the OS

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

# UNIX Philosophy

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 OS - UNIX free

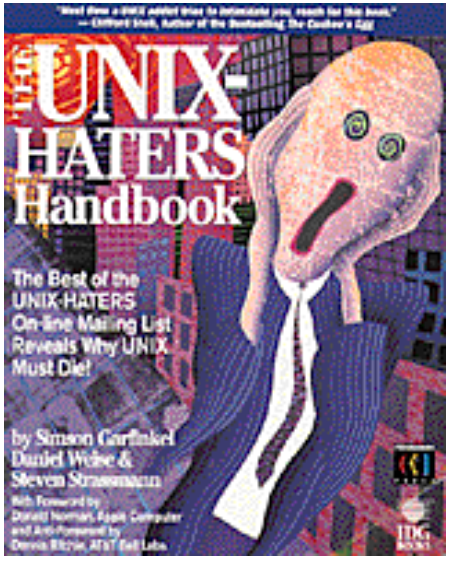
```
From: torvalds@klaava.Helsinki.FI (Linus Benedict Torvalds)
Newsgroups: comp.os.minix
Subject: What would you like to see most in minix?
Summary: small poll for my new operating system
Message-ID: <1991Aug25.205708.9541@klaava.Helsinki.FI>
Date: 25 Aug 91 20:57:08 GMT
Organization: University of 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 is not just Open Source or Gratis

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

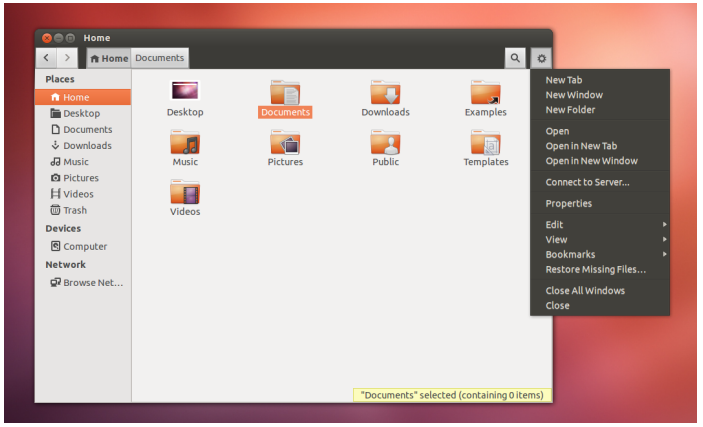
# UNIX architecture



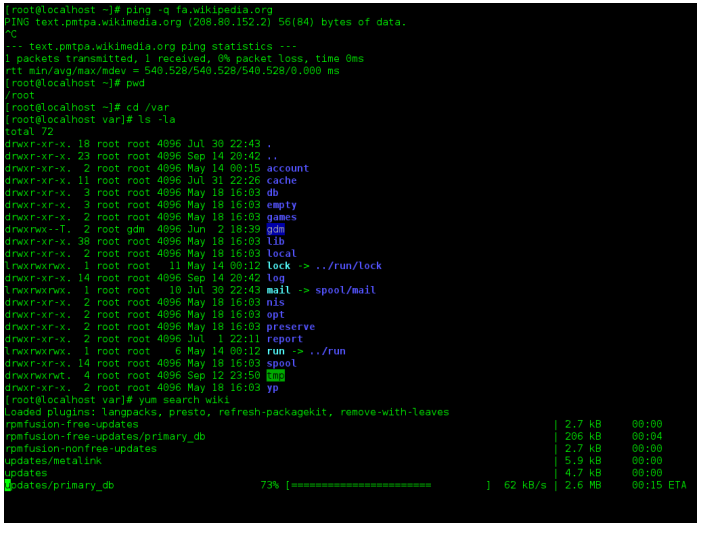

- The kernel is a program, running in supervisor mode, that acts as a program loader and supervisor for the user programs providing locking and I/O services.
- The system services or daemons are programs running on the system and providing facilities that allow or enhance access to system resources
- User programs are task controlled interactively or through batch job submission system by physical user concurrently accessing system resources through multi tasking sharing the CPU(s).

# The Command Line

**Graphical User Interfaces (GUI)**:
comfortable, sometimes lack flexibility




**Command Line Interface (CLI)**:
powerful, requires knowledge




We're here to learn (mostly) how to use CLI:
headaches at first, more productivity in the end

# Authentication

The CLI is THE interface to a UNIX system. To reach the command interpreter a standard procedure is used:

- **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 the command interpreter.

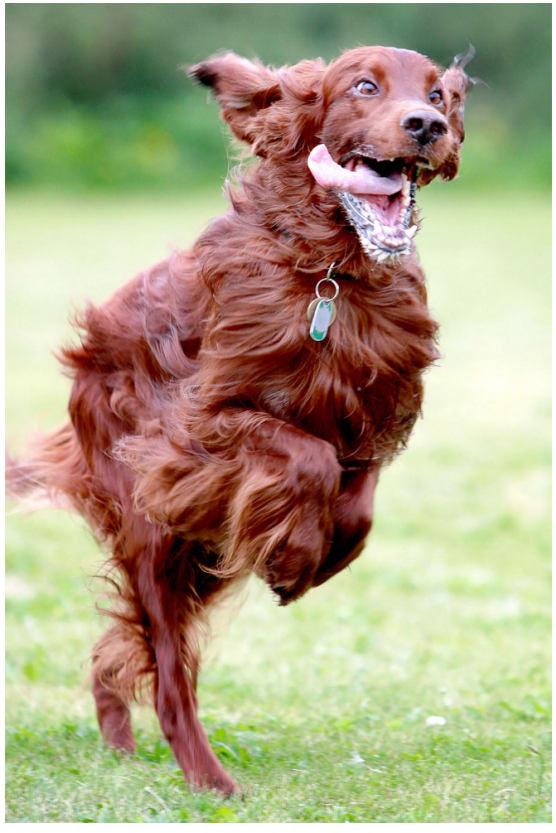
# The command shell

The SHELL is is the 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!

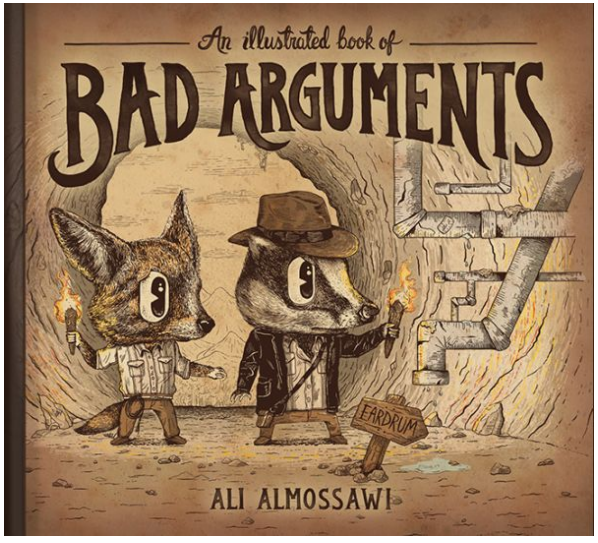
# The Program



A program running in the system has:

- An executable file containing:
  - The instruction for the processor
  - The data to be processed
- The User credentials to access the system resources
- An extendable set of system resources (CPU, Memory, Storage)
- The capability to load system shared executable procedures in the form of routines and functions in libraries.

# Running a program




To run a program in the shell you must know in advance:

- The program path to the executable in the file system or the program name if it is in a system path
- The options that can modify its behavior to fit your requests
- The type and number of arguments the program expects to execute

As an example, these are valid syntaxes to run a program:

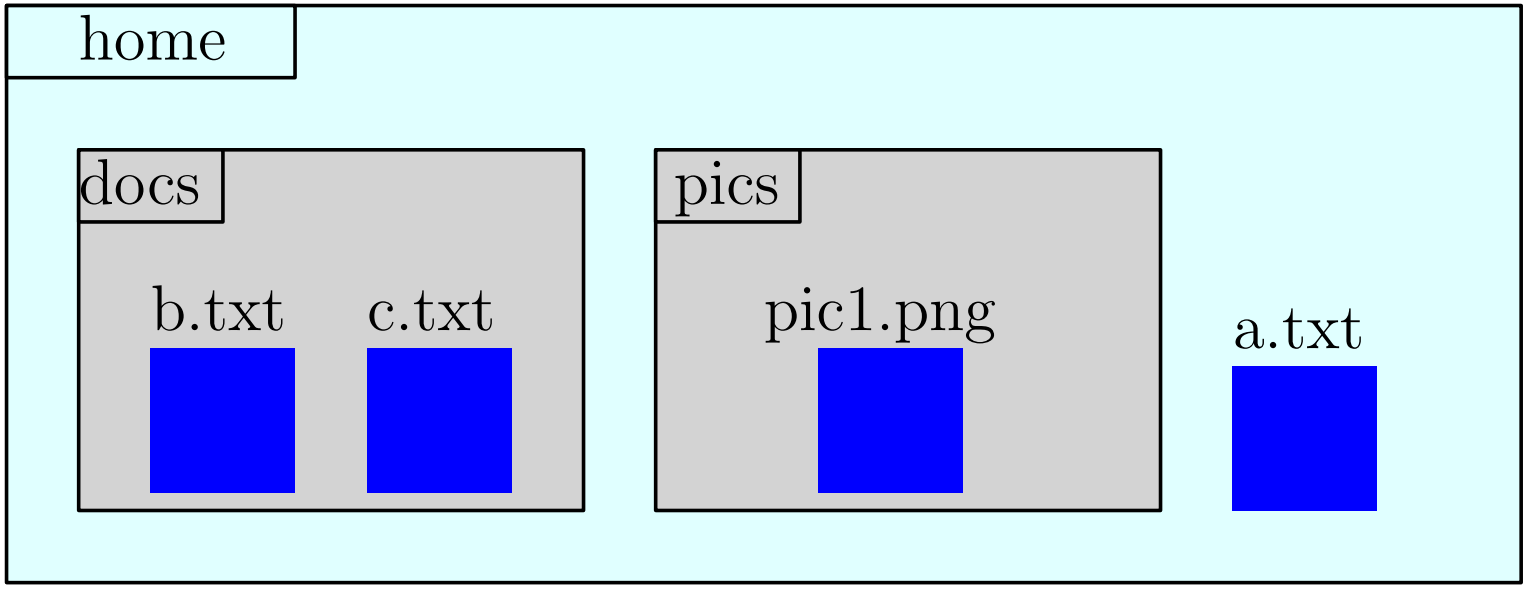
```
ls -l Documents
cp theorem.tex Documents
```

What is a file path in UNIX ?

# Files, directories, exploring the filesystem

**Files contain information, directories contain files**

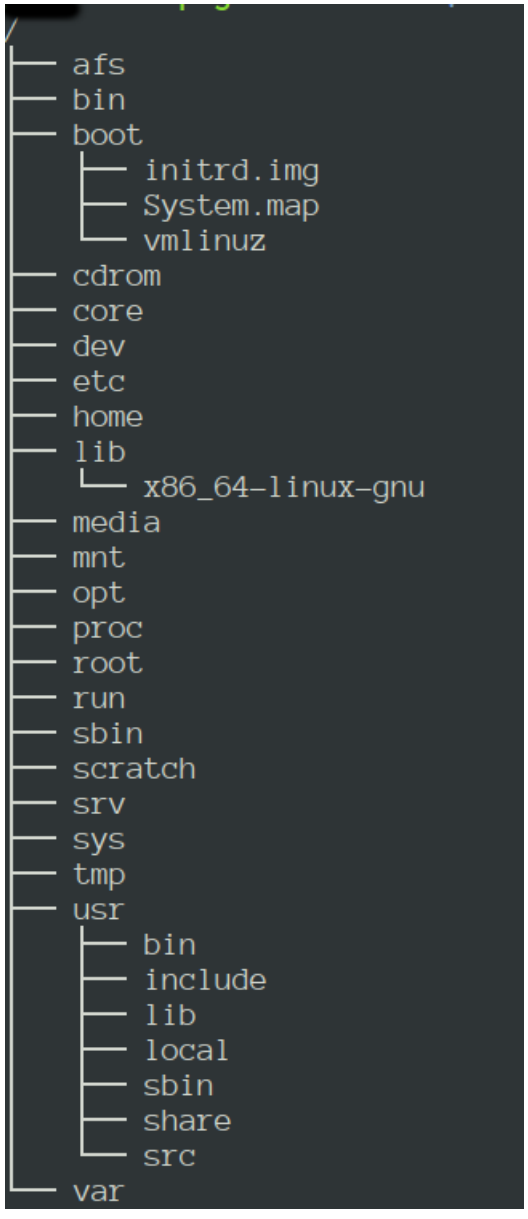
From the terminal, you **navigate** the filesystem,
exploring directories and opening files




Files and directories have a **path** in the filesystem
(for instance, `/home/docs/c.txt`)

Directories in a path are separated by `/`
directory names can end in `/`, files can have an extension (e.g., `.txt`)

# The file system




The Filesystem Hierarchy Standard (FHS) defines the directory structure and directory contents in Linux:

- The filesistem root is the "/" directory
- All files and directories appear under the root directory /, even if they are stored on different physical or virtual devices.
- bin stands for binaries and contains system executables
- lib stands for libraries and contains shared or static pieces of code which are used by running executables or to create executables
- tmp stands for temporary and is where scratch files are placed
- etc stands for etcetera and is where configuration files are
- home is where user files are placed

# Commands I - Pathfinding

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

`pwd`: print the current directory path

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

`pwd` prints the **global** path
(path with respect to the **root directory**)
Example: `/home/documents/text/a.txt`

In commands, you can mostly use the **relative path**
(path with respect to the **current folder**)
Example: `text/a.txt` if I am in `/home/documents/`

# Commands II - Moving around

`ls <directory>`: list the files in the given directory

no argument: list current directory

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

`cd <directory>`: move to another directory

no argument: move to your home directory

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

- `cd ~` brings you to your home directory
- `.` is your current directory
- `..` is the *parent* directory (the directory containing the current one)
- `cd -` brings you to the previously visited directory

# Commands III - Creating and removing

`mkdir <directories>`: create new 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 ~]$
```

`touch <filenames>`: create new (empty) text files

```
[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 ~]$
```

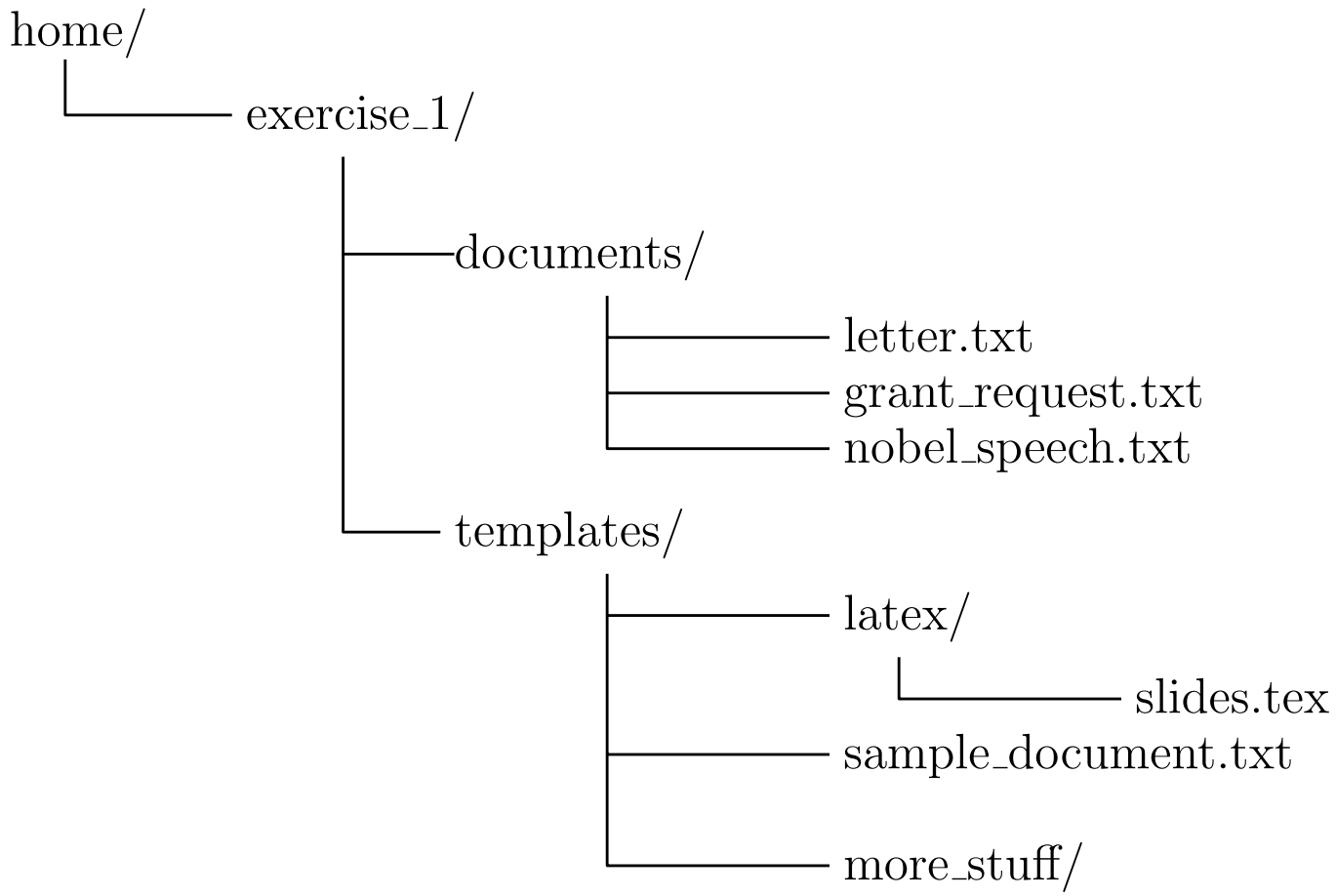
`rm <filenames>`: removes files

`rm -r <directories>`: removes 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 ~]$
```

# Exercise I - mkdir, rm

Using the commands you know, create these directories and files, and then remove them all (**after showing us**):



# Commands IV - Moving and copying files

`cp <old_path> <new_path>`: **copy** a file to another location

`cp -r`: copy entire directories

```
[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 ~]$
```

Paths can be relative, the copy can have a different name

```
[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]$
```

`<new_path>` is overwritten, old content is lost

Suggestion: use `cp -i`

`mv`: same syntax, target **moved** (old file/directory deleted)

# Commands V - Finding and listing files

`find` **recursively searches files in a directory**

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

Many options, we will see the basic ones:

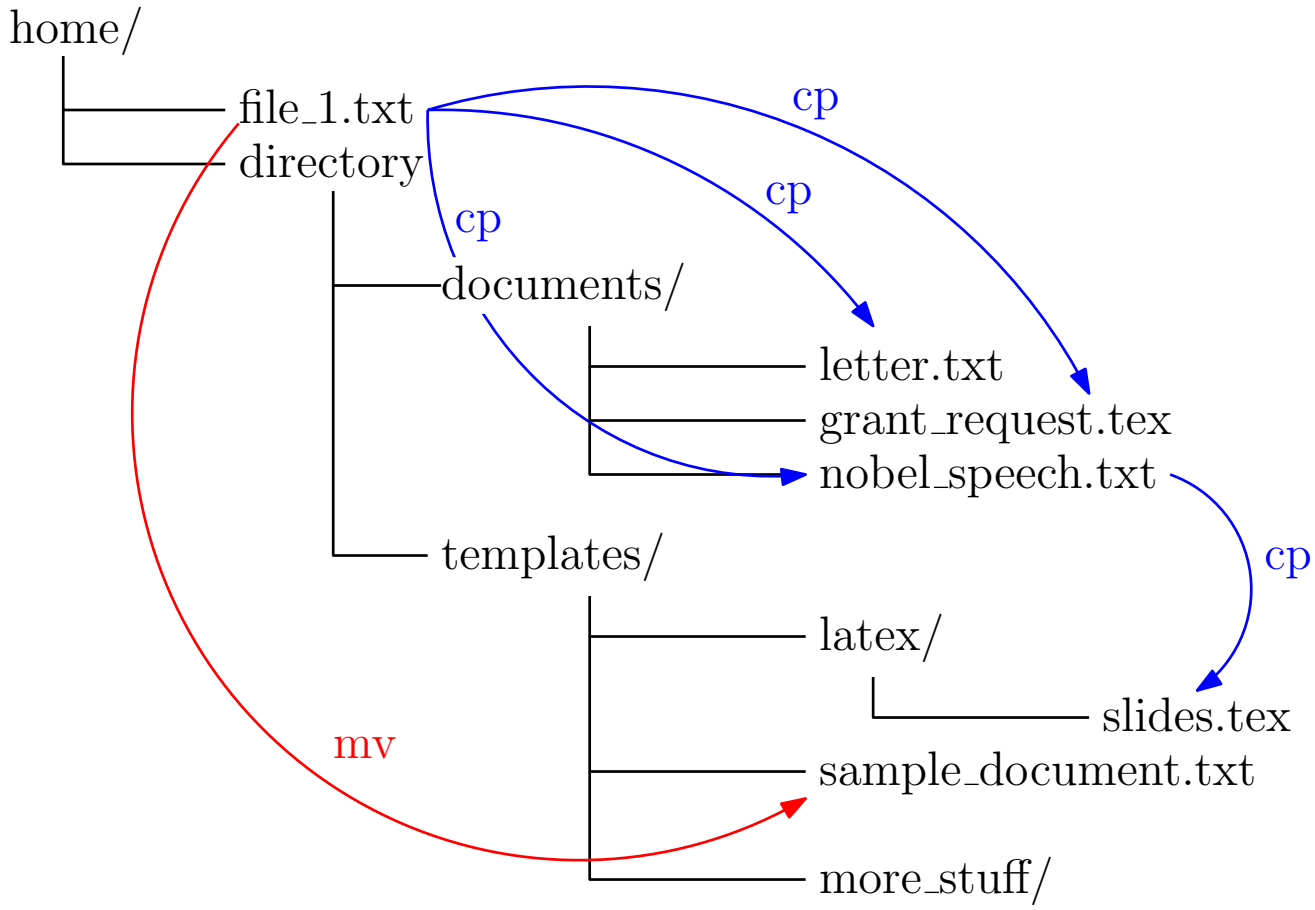
- `-name`:
  specify file name (no paths here)
- `-path`:
  specify (part of) the file path
- `-printf %{format}`:
  print details of the items found
- `-delete`:
  deletes the files found

```
~/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
---------------------------------------
```

In the above commands, 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