

Shell commands

Terminals and shells

Terminal

- **Terminal**: a text-based interface with keyboard input
 - Can be a physical device



- or a graphical application (a **terminal emulator**).

Shell

- **Shell**: reads commands from its input and executes them
 - An **interactive** shell runs in a terminal.
 - Alternatively, can read commands from a file (a **shell script**).

Commands

Shells are a type of *command-line interface*.

We use them by entering commands, with their arguments:

```
command arg1 arg2 arg3
```

Example:

```
sleep 1.5
```

Comments

Everything after “#” is ignored:

```
# The following command waits for 1.5 s  
sleep 1.5
```

Documentation

Most commands respond to the `--help` argument with a short usage message:

```
sleep --help
```

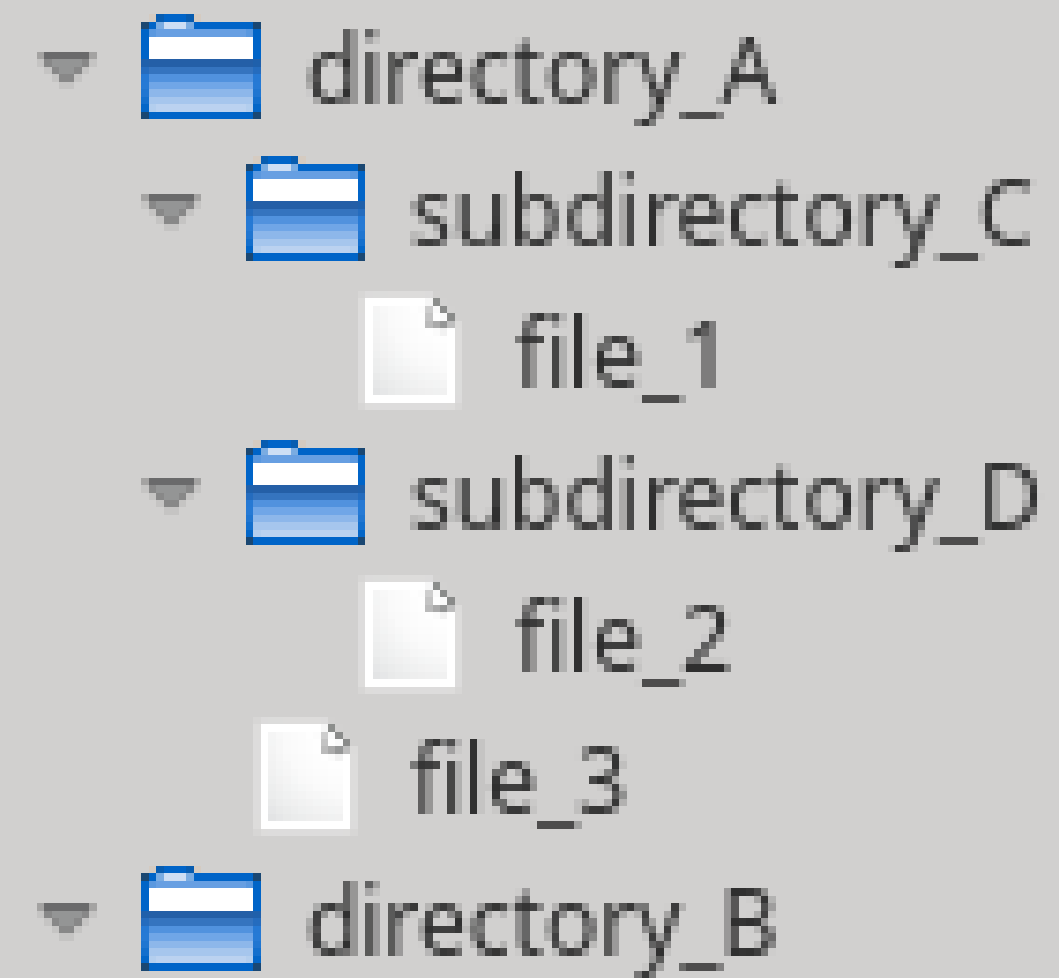
Most commands also have an extensive manual. To consult it, use the `man` command:

```
man sleep
```

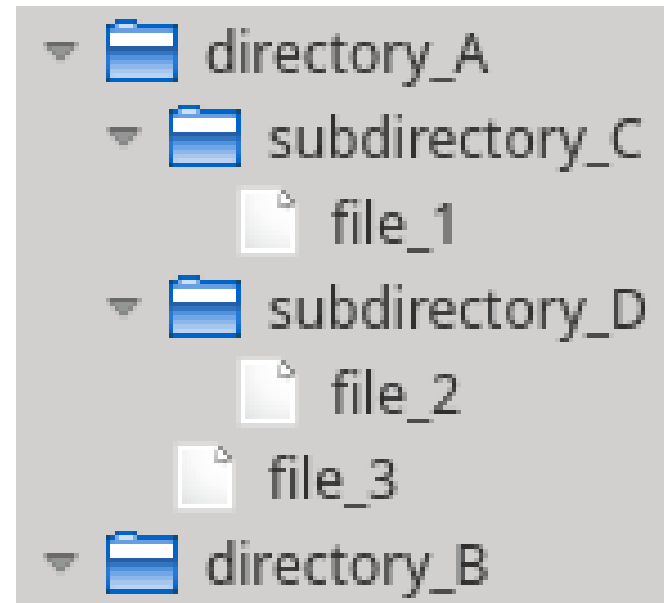
Shell keyboard shortcuts

- **Control+C**: Interrupt the currently command
- **Up/Down**: Browse command history
- **TAB**: Complete command as far as unambiguously possible
- **TAB (again)**: In case of ambiguity, list possible completions

Filesystem



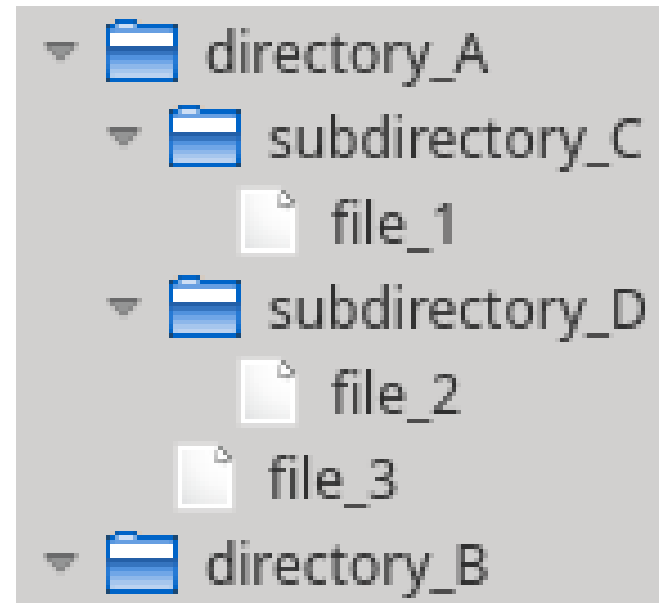
Absolute path



An **absolute path** starts with a /

Example: /directory_A/subdirectory_D/file_2

Special directories



Two special subdirectories are present in every directory:

- “.” stays in the same directory
- “..” goes to the parent directory

Example: the following are equivalent absolute paths:

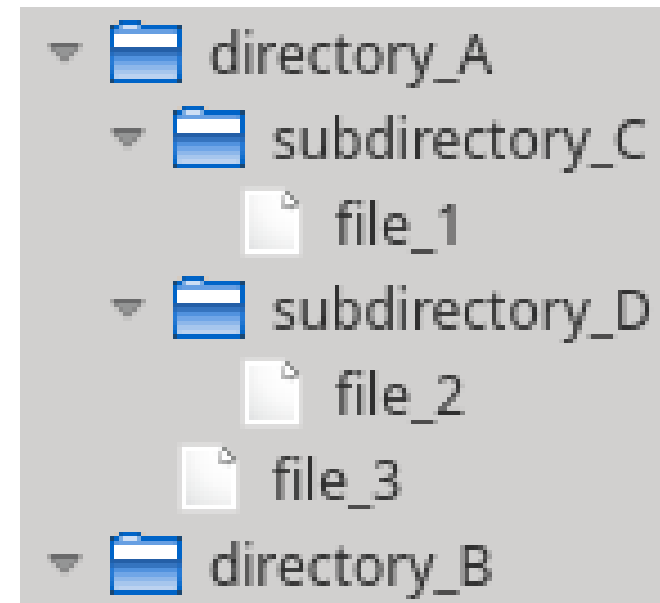
- `/directory_A/subdirectory_D/file_2`
- `/directory_A/subdirectory_C/../../subdirectory_D/file_2`

- `/directory_A/../../subdirectory_D/../../file_2`

Current directory

All processes (incl. shells) have a **current working directory** that can change over time.

Any path **not** starting with / is **relative** to the current working directory.



Example: If the current working directory is

`/directory_A/subdirectory_D`

then the following are equivalent paths:

- `/directory_A/subdirectory_D/file_2`
- `file_2`
- `../subdirectory_D/file_2`
- `../subdirectory_C/../../subdirectory_D/file_2`

Commands for navigating the filesystem

`pwd` Show the working directory

`ls` List files – by default, prints the contents of the current directory to standard output

`cd` Change the current directory

Commands for viewing files

<code>cat</code>	Show (“print”) the contents of files on standard output
<code>less</code>	Display the contents of files (allows browsing them with Up/Down) – Type q to exit
<code>hexdump</code>	Print the contents of files in hexadecimal
<code>strings</code>	Print the parts of files that are printable (ASCII) characters
<code>objdump</code>	Print information about object, shared object, and executable files

Commands managing files

<code>cp</code>	Copy a file
<code>mv</code>	Move (rename) a file
<code>rm</code>	Remove (delete) a file
<code>mkdir</code>	Create (make) a directory
<code>rmdir</code>	Remove an empty directory

The most important command

`man` Documentation for all the above commands

“Globbing”

Globbing: simple pattern matching to match file paths

- ***** matches any string of characters incl. empty string
- **?** matches exactly one character.
- Characters between **[]** match exactly one of the specified characters.
- Character ranges can be specified using **-**

pattern	matches				
a*z	abcz	a00z	az	...	
a??	abc	a00	aaa	...	but not: ab
a[0123].jpg	a0.jpg	a1.jpg	a2.jpg	a3.jpg	
a[0-3]	a0	a1	a2	a3	
a[b-d]	ab	ac	ad		

Other useful commands

Utilities

echo	Print command-line arguments to standard output
top	Display currently-running processes – Type q to exit
htop	Alternative to top
time	Run commands passed on the command line, measure the time they take to run
touch	Update the last-modified time of a file, create it if it does not exist
head	Print the beginning of a file
tail	Print the end of a file

Download, decompress

wget	Download files from the internet
curl	Download files from the internet
zip	Compress files in the zip format
unzip	Decompress files in the zip format
tar	Create and extract archives in the tar (and tgz) format

Access rights, sysadmin

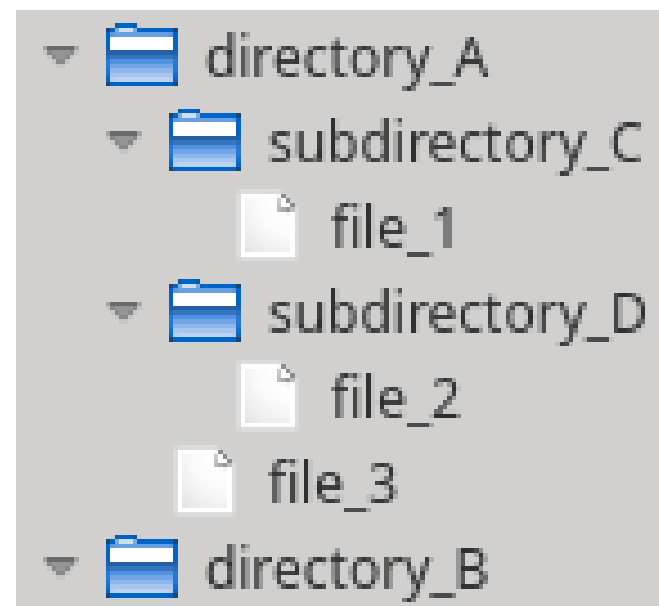
chmod	Change the access rights (read, write, execute) of a file
chown	Change the ownership (user and group) of a file
su	Run a shell as root (superuser / administrator)
sudo	Run commands passed on the command line as root

The most important command

`man` Documentation for all the above commands

Running executables

If a command contains a `/`, the shell interprets it as a path and tries to **execute** it.



Example:

```
# make "file_3" executable and run it  
cd /  
cd directory_A  
chmod +x file_3  
./file_3
```

Redirections

Every process has

- a standard input (“stdin”, by default keyboard input)
- a standard output (“stdout”, by default a terminal display)

We can **redirect** stdin and stdout with `<` and `>`.

Examples:

```
# Write the current directory's file list to a file called "list.txt":
```

```
ls > list.txt
```

```
# Print the content of "list.txt"
```

```
cat < list.txt
```

We can **pipe** the stdout of a command to the stdin of another.

Example:

```
# Print the content of "list.txt" in hexadecimal, pipe to the "less" pager:  
hexdump list.txt | less
```

SSH

SSH (Secure SHell) allows connecting to a remote machine.

The usage is

```
ssh user@host
```

where host is a numeric IP address like 192.168.0.100 or a host name like host.example.com

After authentication (password or public key), we get a **remote** shell. All commands are run on the remote machine.

Copying files

SCP uses the SSH protocol to copy files.

The usage is

```
scp user@host:remote_path local_path
```

```
scp local_path user@host:remote_path
```

Public-key authentication

In your user's ("home") directory, there should be a `.ssh/` subdirectory. It should contain `id_*` and `id_*.pub`.

Typically:

- `id_rsa` and `id_rsa.pub` or
- `id_ed25519` and `id_ed25519.pub`

If not, run `ssh-keygen` to create them.

The `.pub` file contains a **public** key. You can freely share it with anyone.

The other file contains the corresponding **private** key. It is secret and you **must not share it**.

When connecting to an ssh server:

- It gives you a challenge that can only be solved by knowing your **private** key.
- Anyone knowing your **public** key can **check** that you correctly solved the challenge.