

CS17

Terminal & Shell

A practical, printable lesson plan for becoming confident with the command line and filesystem.

Starting point	Students can use application software, understand files/folders/extensions, and have used MIT Scratch.
End goal	Students can confidently navigate the filesystem, manipulate and search files, combine shell commands, and recover from common errors.
Teaching approach	Mental models first. Commands second. Repeated hands-on practice instead of memorisation.

Core idea

“I can control the same computer through clicks or through typed instructions.”

Recommended sequence: 5 lessons, followed by one practical assignment. The focus is terminal literacy, filesystem confidence, and safe command-line habits.

Module overview

The terminal should feel like a new interface to concepts students already know: folders, files, applications and actions. The biggest conceptual target is the filesystem: where am I, what path am I using, and what will this command act on?

Lesson	Focus	Key ideas
1	Talking to the computer	Terminal vs shell, pwd, ls, cd, paths, mkdir
2	Files without a GUI	touch, cat, echo, cp, mv, rm, arguments, flags
3	Becoming fast	Tab completion, history, spaces, wildcards, grep, wc
4	Combining commands	Input/output, pipes, redirection
5	Help, errors & safe habits	--help, man, reading errors, case sensitivity, deletion safety

Teaching rule

Do not begin with a giant cheat sheet. In Lesson 1, repeatedly use pwd, ls, cd, mkdir and .. until students understand “where am I?” and “where am I going?”.

Assessment philosophy

- Ask “When would you use this?” more often than “What does this stand for?”
- Give actions in plain English and let students choose commands.
- Let students make mistakes and read the terminal error before helping.
- Reward correct mental models even when a command is mistyped.

LESSON 1

Talking to the Computer

Suggested duration: 60-75 minutes

By the end, students should be able to:

- Explain terminal vs shell in simple terms.
- Identify the current working directory.
- Move through folders using relative paths.
- Create a directory from the shell.

1. Start from something familiar

Create a folder called CS17 on the Desktop using the normal graphical file manager. Then open Terminal and say: “We are going to make the computer do the same thing without touching the mouse.”

2. Terminal vs shell

- Terminal = the application/window we type into.
- Shell = the text-based interface that reads our commands and asks the operating system to carry them out.

Keep the distinction light. At this stage, the important point is that the terminal gives us a text interface to the same computer.

3. First commands

```
pwd  
ls
```

pwd tells us where the shell is currently “standing”. ls lists what is there. Open the same location in Finder/File Explorer and compare the names.

4. Move around

```
cd Desktop  
pwd  
ls  
cd ..  
cd ~
```

.. means the parent directory. ~ means the user’s home directory.

5. Paths: the main mental model

```
Home
|-- Desktop
|-- Documents
|   |-- CS17
|       |-- notes.txt
|       |-- homework.txt
|-- Downloads
```

Absolute path = full address. Relative path = directions from where you currently are.

- Absolute analogy: a complete postal address.
- Relative analogy: “from school, take the second left.”

6. Create a folder

```
mkdir cs17-terminal
ls
cd cs17-terminal
pwd
```

Stop here. A short set of commands understood deeply is more valuable than a long list memorised poorly.

In-class challenge

- Go to Desktop.
- Create terminal-practice.
- Enter it, then create lesson-1.
- Enter lesson-1 and print your current location.
- Return to terminal-practice.

Homework

Draw a folder tree on paper and write commands for moving between different branches. Include at least one use of `..` and one relative path.

LESSON 2

Files Without a GUI

Suggested duration: 60-75 minutes

By the end, students should be able to:

- Create and inspect text files.
- Copy, move, rename and delete files.
- Explain command, argument and option.
- Understand why command-line deletion needs care.

Warm-up

Students should use these from memory: `pwd`, `ls`, `cd`, `cd ..`, `mkdir`. Ask for use-cases, not expansions.

1. Create and inspect files

```
touch hello.txt
ls
cat hello.txt
```

Point out that `.txt` is part of the filename. The command created a file; the extension simply helps identify its intended type.

2. Produce text and put it in a file

```
echo "Hello CS17"
echo "Hello CS17" > hello.txt
cat hello.txt
```

For now: `echo` produces text and `>` sends that output into a file. Formal input/output comes later.

3. Copy, rename and move

```
cp hello.txt copy.txt
mv copy.txt message.txt
mkdir backup
mv message.txt backup/
```

`mv` can both rename and move because both operations change the file's name or location.

4. Delete carefully

```
rm hello.txt
rmdir empty-folder
rm -r folder-name
```

Explain that terminal deletion often does not have the friendly Trash/Undo workflow students are used to. Introduce recursive deletion only after they understand that folders can contain other items.

5. Command anatomy

```
cp hello.txt backup.txt
```

command argument argument

Mental model: command = what to do; argument = what to do it to; option = how to do it.

6. Options / flags

```
ls
ls -l
ls -a
```

Compare the output visually and let students infer what changes.

Practice challenge

```
project/
|-- images/
|-- notes/
|   |-- idea.txt
|   |-- todo.txt
|-- backup/
```

- Create the structure.
- Copy idea.txt into backup.
- Rename todo.txt to tasks.txt.
- Delete the original idea.txt.

LESSON 3

Becoming Fast With the Terminal

Suggested duration: 60 minutes

By the end, students should be able to:

- Use completion and history instead of retyping.
- Handle filenames containing spaces.
- Use wildcards to operate on groups of files.
- Search text and count lines.

1. Tab completion

```
cd Doc<Tab>
```

Create deliberately long folder names and encourage students to use Tab. Good terminal users do not type everything.

2. Command history

Up arrow / Down arrow

```
history
```

```
clear
```

Show how quickly previous commands can be edited and rerun.

3. Spaces in filenames

```
cd CS17 Notes    # fails: shell sees two arguments
```

```
cd "CS17 Notes"
```

```
cd CS17\ Notes
```

This is a gentle introduction to quoting and escaping. Keep it practical.

4. Wildcards

```
photo1.jpg
```

```
photo2.jpg
```

```
photo3.jpg
```

```
notes.txt
```

```
ls *.jpg
```

```
ls photo*
```

```
cp *.jpg images/
```

* means “match anything here”. Ask students to predict matches before pressing Enter.

5. Search inside text

```
grep apple fruits.txt  
grep mango fruits.txt
```

Do not teach regular expressions yet. The important idea is simple: a command can search through a text file.

6. Count

```
wc fruits.txt  
wc -l fruits.txt
```

`wc -l` is especially useful because it counts lines.

Quick drill

- List only .txt files.
- Copy all .jpg files into images/.
- Search notes.txt for the word Scratch.
- Count the number of lines in students.txt.

Teacher checkpoint

Before moving on, students should be able to explain why `cd "CS17 Notes"` works but `cd CS17 Notes` does not.

LESSON 4

Combining Commands

Suggested duration: 60-75 minutes

By the end, students should be able to:

- Explain input and output at a basic level.
- Pipe output from one command to another.
- Redirect output into a file.
- Compose small commands into a larger solution.

1. Start simple

```
cat fruits.txt
```

```
cat fruits.txt | grep apple
```

```
cat fruits.txt | grep apple | wc -l
```

Explain | as: “take the output from the command on the left and give it to the command on the right.” Always ask students to predict the result before running the command.

2. General input/output model

```
INPUT  ->  COMMAND  ->  OUTPUT
```

```
fruits.txt -> grep -> matching lines
```

This model is useful for understanding how information moves through shell commands.

3. Redirection

```
ls > files.txt
```

```
cat files.txt
```

```
echo "Mango" >> fruits.txt
```

> writes/replaces. >> appends.

4. Composition

Show that each command can do one small job, and the shell lets us connect those jobs together.

```
cat fruits.txt | grep apple | wc -l
```

Main challenge

Create a `students.txt` file with repeated names. Ask students to find how many lines contain “Ali”, then save only the count into `ali-count.txt`.

```
grep Ali students.txt | wc -l > ali-count.txt
```

Reveal the solution only after students have attempted to compose it themselves.

Discussion prompts

- What is the output of `grep Ali students.txt`?
- What does `wc -l` receive as its input?
- Where does the final number go after `> ali-count.txt`?
- What would change if we used `>>` instead?

Key mental model

The shell can connect simple commands so that the output of one becomes the input of another.

LESSON 5

Help, Errors & Safe Habits

Suggested duration: 60 minutes

By the end, students should be able to:

- Use built-in help before asking for the answer.
- Read common error messages calmly.
- Recognise case-sensitive filenames and path mistakes.
- Develop safe habits around destructive commands.

1. Ask the command for help

```
ls --help
cp --help
rm --help
```

On many systems, commands provide a `--help` option. Students do not need to read everything - teach them to scan for the option or behaviour they need.

2. Manual pages

```
man ls
man cp
```

If `man` is available, show how to scroll and press `q` to quit. Treat documentation as something normal to consult, not something to memorise.

3. Errors are information

```
cd folder-that-does-not-exist
cat missing.txt
cp missing.txt backup/
```

Teach a repeatable response to failure:

- Read the whole error.
- Look at the command you typed.
- Check the current directory with `pwd`.
- Check filenames with `ls`.
- Try again with the corrected path or name.

4. Case sensitivity

```
touch Notes.txt
cat notes.txt
```

Depending on the filesystem, Notes.txt and notes.txt may be treated as different names. Use this to teach careful reading rather than memorising platform-specific rules.

5. Safety with deletion

Before using `rm` or `rm -r`, build the habit of checking exactly what will be affected.

```
pwd
ls
# check the target carefully
rm unwanted.txt
```

Do not introduce `rm -rf` as a normal beginner shortcut. The goal is confidence plus caution.

6. A useful troubleshooting routine

1. Where am I? -> `pwd`
2. What is here? -> `ls`
3. Is the name exact? -> check spelling/case/spaces
4. What does help say? -> `command --help`
5. Try again.

End-of-module habit

A confident terminal user is not someone who never makes mistakes. It is someone who can inspect the situation, understand the error, and recover.

Final practical assignment

The Lost Research Project

Prepare this folder in advance:

```
research-project/  
|-- notes.txt  
|-- students.txt  
|-- image1.jpg  
|-- image2.jpg  
|-- experiment.txt  
|-- old-notes.txt  
|-- report.txt  
|-- data/  
    |-- result1.txt  
    |-- result2.txt  
    |-- result3.txt
```

Students must complete the following using only the terminal:

- Create folders named images, notes and archive.
- Move all JPG files into images.
- Move old-notes.txt into archive.
- Rename notes.txt to research-notes.txt.
- Copy report.txt into archive.
- Find every line containing success inside experiment.txt.
- Count how many lines contain success.
- Save the count into success-count.txt.
- Display success-count.txt.
- Print the current working directory.

Teacher note

Do not provide a one-to-one command list. The assignment is intended to test whether students can translate a desired action into commands and combine what they know.

End-of-module reference

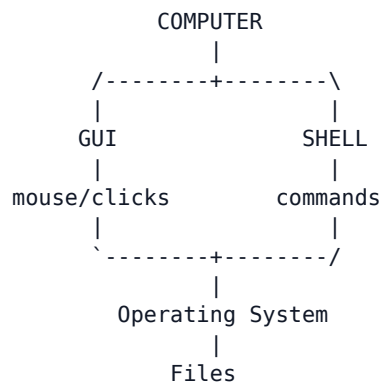
Commands students should know

Command	Use
pwd	show current directory
ls	list items
cd	change directory
mkdir	make directory
touch	create empty file
cat	display file contents
cp	copy
mv	move / rename
rm	remove file
rmdir	remove empty directory
echo	produce text
clear	clear terminal display
history	show command history
grep	search text
wc	count lines / words / bytes
man	open a manual page (where available)

Operators & shortcuts

>	write output to a file
>>	append output to a file
	send output to another command
*	wildcard match
..	parent directory
~	home directory
Tab	autocomplete
Up / Down	command history
Ctrl + L	clear screen

Final mental model



The aim is for students to see the shell as another way of interacting with the same filesystem they already understand through folders and applications.

Success condition

The terminal no longer feels mysterious. Students can move around confidently, manipulate files, search text, combine simple commands, and recover from common mistakes without needing a graphical interface for every action.

Good stopping point

At this stage, stop. Do not add package managers, permissions, SSH, environment variables, or advanced Unix administration topics until a later module creates a real need for them.

This keeps the module focused on one skill: being comfortable with the terminal itself.