# Core Terminal Commands

*AiCE Software Engineering Foundations — The Developer Environment*

Now that you understand what a terminal is and how its prompt describes your location, it's time to actually move around and start creating things. This lesson walks through the small set of commands you will type more often than any others: finding where you are, seeing what's around you, moving between folders, and creating your first files.

## 1. Finding Where You Are: pwd

At any moment in the terminal, you are located inside a specific directory — this is called your working directory. To find out exactly which directory you are currently in, type:

```
pwd
```

pwd stands for “print working directory.” It will print the full path of your current location, for example:

```
/home/yourname
```

That particular path is your home directory — the directory that belongs to your user account. Think of it as your personal office: everything you create lives here, or in subdirectories underneath it.

## 2. Listing Directory Contents: ls

To see what's inside your current directory, use:

```
ls
```

This prints the names of all non-hidden files and directories in your current location. Directories typically appear in blue, files in white — a color code that lets you tell the two apart at a glance.

If you also want to see hidden files, add the -a flag:

```
ls -a
```

> **Hidden Files**
> *In Linux, any file or directory whose name begins with a dot (.) is hidden. Hidden files usually hold system or application configuration, and are left out of a plain ls listing because most users never need to touch them directly.*

## 3. Moving Around: cd

To move into a directory, use cd followed by its name:

```
cd MyCodes
```

After pressing Enter, your prompt updates to reflect the new location — the last segment before the dollar sign changes from ~ to MyCodes, confirming exactly where you now stand.

To go back up one level, use the special double-dot symbol, which always represents the parent directory (the directory that contains your current one):

```
cd ..
```

Think of cd .. as the “back” button in a file manager. If you were inside /home/yourname/MyCodes and ran cd .., you would land back in /home/yourname.

## 4. Creating Directories: mkdir

To create a new directory, use mkdir followed by the name you want to give it:

```
mkdir AfricaTechHub
```

Run ls afterward to confirm it now exists inside your current directory.

**Naming Rule**

*Never use spaces in file or directory names. If a name needs multiple words, join them with an underscore or a hyphen instead — for example nairobi_startups, lagos-dev-bootcamp, or KigaliCodeAcademy. All three styles are acceptable; the important thing is to pick one and stay consistent.*

## 5. Creating Files: echo

Directories organize things, but files are where actual content lives — your code, documents, and data. One of the simplest ways to create a file directly from the terminal is the echo command:

```
echo "Hello, Africa!" > welcome.txt
```

This writes the text “Hello, Africa!” into a file called welcome.txt, creating the file if it doesn't already exist. The greater-than sign (>) is called output redirection — instead of printing echo's output to the screen, it routes that output into a file.

**File Extensions**

*The .txt at the end of welcome.txt is a file extension — it signals what kind of content the file holds. Text files use .txt, Python files use .py, JavaScript files use .js. Unlike a graphical interface, the terminal never adds an extension automatically; you must type it yourself.*

To create a file with content spanning multiple lines, use echo -e together with \n to insert line breaks:

```
echo -e "Africa is rising.\nTechnology is the engine.\nWe are
the
engineers." > manifesto.txt
```

Each \n means “start a new line here.” Be careful not to put a space right after \n, or the next line will begin with an unwanted blank space.

## 6. Reading Files: cat

cat technically stands for “concatenate,” but its most common everyday use is simply printing a file's contents to the screen. The syntax is:

```
cat filename.txt
```

For example, running cat welcome.txt would print:

```
Hello, Africa!
```

> **A Note on Long Files**
> *cat prints an entire file at once, which becomes hard to read for long files. For those cases, commands like less or more allow paginated reading. For short files, though, cat remains the fastest option.*

## Key Takeaways

With just six commands — pwd, ls, cd, mkdir, echo, and cat — you can already answer the three questions every terminal session revolves around: where am I, what's here, and how do I create or read something. These commands will appear constantly in every later unit of this course, so the goal is not to memorize them from a page, but to type them enough times that they become automatic.