Where I Am Now

AUTHOR Mohammed Efaz | IDEA Roadmap | WORDS 820 | READ TIME ~5 min

Where I am now

This is the page I need for ordinary study sessions. I can return to the full route, the 48-book audit, or the rules that keep the plan mine when I need the reasoning behind the next action.

My current stage

I am in **Stage 1: C, shell, Python DSA, and Unix context**.

1. C Programming: A Modern Approach, 2nd Edition

I am at **Chapter 6: Loops**. I have implemented and verified all 12 Chapter 6 programming projects. My next job is to review the chapter's ideas and solve the exercises independently before I continue.

While the book is teaching C99-compatible material, I will compile with strict warnings:

```
cc -std=c99 -Wall -Wextra -Wpedantic -g program.c -o program
```

When pointers, arrays, or dynamic memory enter the work, I will use sanitizers:

```
cc -std=c99 -Wall -Wextra -Wpedantic -g \
  -fsanitize=address,undefined program.c -o program
```

Warnings are part of the lesson. I should be able to explain every one instead of silencing it and hoping for the best.

2. The Linux Command Line, 3rd Edition

I will continue from the beginning or from my current bookmark. I need to type the commands rather than read them passively, and I will use a disposable practice directory while I learn.

The shell is also where the C work becomes real. I will use it to compile, run, inspect, and debug the programmes from King.

3. A Common-Sense Guide to Data Structures and Algorithms in Python, Volume 1

I will continue from Chapter 1 or from my current bookmark. I will implement the important structures and algorithms in Python, test them, and state their time and space complexity.

Learning C elsewhere does not mean I need to drag C's memory-management cost into DSA practice. Python remains my main DSA language.

4. UNIX: A History and a Memoir

This is the lighter book in the stage. I can read it whenever I want a change of pace. It does not need exercises, notes, or a project.

My next session

If I sit down and do not know which book to open, I will do this:

1. Open King at Chapter 6.
2. Choose 1 loop concept or exercise.
3. Predict the programme's behaviour before compiling it.
4. Implement it without looking at a finished answer.
5. Compile with warnings enabled.
6. Test normal, boundary, and invalid input where it matters.
7. Explain the loop condition, the changing state, and the termination rule.
8. Leave an exact section, page, or exercise marker for the next session.

If I need a change of pace, I can use *The Linux Command Line* or DSA Volume 1. Difficulty in the current work is not a reason to escape into a future-stage book.

The study loop

```
read -> predict -> implement -> compile/run -> test -> inspect -> explain -> co
```

- **Read:** I need the explanation before I start copying syntax.
- **Predict:** I should decide what I expect before the compiler gives me an answer.
- **Implement:** I turn the prose into a programme, command, or experiment.
- **Compile and run:** I expose syntax, type, and environment problems.
- **Test:** I check more than the sample input.
- **Inspect:** Later this will include debuggers, sanitizers, traces, disassembly, profilers, and hardware registers.
- **Explain:** I make sure the success was not accidental.
- **Continue:** I leave a precise marker so the next session starts cleanly.

What completion means

- For a textbook, I will read the main teaching material, run useful examples, and do enough exercises to use the ideas independently.
- For a project book, I will build the projects and make small changes of my own.
- For a pocket guide or manual, I will make a structured pass, try unfamiliar material, and keep it for lookup.
- For a security book, I will work only inside systems I own or have explicit permission to test.
- For a memoir, I will read it normally.

I do not need to complete exercises that repeat an idea I already understand, create elaborate notes, or manufacture proof that I studied. I should not skip an exercise or project that introduces something new.

When I can leave Stage 1

I will move to Stage 2 when all of these are true:

- I have completed King's main teaching material from my current position and the non-repetitive projects that introduce new ideas.
- I can write, compile, test, and debug small multi-file C programmes from the terminal without copying a template.
- I am comfortable with control flow, functions, arrays, pointers, strings, structures, files, scope, storage duration, and the common undefined behaviour covered by King.
- I have worked through *The Linux Command Line* and used the commands rather than only reading them.
- I have finished DSA Volume 1 with working Python implementations of its central structures and algorithms.
- I have finished the Unix memoir.

When something blocks me

I will give the blocker 1 focused attempt and preserve the exact error message. If it remains unresolved, I can switch to another current-stage book and return later. I will not lose days installing an environment for work that belongs several stages ahead.

Kernel, embedded, Kali lab, CUDA, and model-performance environments can wait until I reach them.