How I Arrived at This Roadmap

AUTHOR Mohammed Efaz | IDEA Roadmap | WORDS 613 | READ TIME ~4 min

How I arrived at this roadmap

The current plan came from 2 earlier attempts. Git preserves both versions, but I no longer keep them as competing folders because I want 1 route I can follow without reopening the same decision every week.

The first attempt: a broad 28-book route

My first roadmap tried to hold the full range of subjects that drew me in:

- C
- Linux command-line work
- Python DSA
- modern C++
- Linux system and kernel programming
- embedded C
- offensive and defensive security
- CUDA
- LLMs
- reasoning models

It reflected what I wanted to learn. It kept Python DSA, security, and reasoning models in the main sequence, and it let me move among a few books inside each stage.

The gaps became clearer when I looked at the route as preparation for real AI systems work. It did not give me a complete practical ML and deep-learning foundation before LLM and inference work. It also blurred the difference between CPU asynchrony, CPU data parallelism, CUDA kernels, and full-stack inference performance. Some broad references were treated as if every selected book needed the same cover-to-cover reading method.

The second attempt: a low-level performance route

The second roadmap narrowed the route towards C, systems, embedded work, C++, machine learning, CUDA, and AI performance.

This version fixed several technical problems. It added practical ML and deep-learning foundations, improved the dependency chain into CUDA, separated Python orchestration from C++ runtimes and GPU kernels, and treated manuals as references rather than ordinary textbooks. It also forced me to audit all 48 local candidates instead of stopping at the books I had already noticed.

The technical order improved, but the plan stopped fitting the person who had to follow it.

It made *Algorithmic Thinking* my main DSA route even though I had chosen Python and disliked the book's selective structure as a primary course. It made *Computer Systems: A Programmer's Perspective* mandatory even though I had already rejected its presentation. It also pushed security and reasoning-model work outside the default path to shorten the route towards performance.

That version optimised the abstract destination and ignored too much of what would keep me moving.

The route I am following now

The current roadmap keeps the useful parts of both attempts.

From the broad route, I kept:

- Python DSA through both Wengrow volumes
- LeetCode 75 in Python
- security in the main sequence
- reasoning models and RLHF as the endpoint
- a simple stage-based way of working

From the performance route, I kept:

- full practical ML and deep-learning foundations

- modern C++ before CUDA
- a clear separation between userspace, kernel, embedded, CPU async, CUDA, and inference performance
- selective use of manuals and references
- the full 48-book audit
- measurement as the centre of the performance stages

I also corrected the choices that did not fit me:

- *Algorithmic Thinking* is optional.
- CS:APP is not selected.
- Owning a book does not make it mandatory.
- Security and reasoning models remain inside the default route.
- The repository contains 1 default roadmap.
- Content-production work stays outside this repository.

The shape of the plan

```
C + shell + Python DSA
-> repeated building
-> modern C + Linux internals
-> C++ + Linux userspace APIs + security Linux
-> C++ projects + deep Linux APIs + Kali methodology
-> memory + kernel + embedded
-> larger asynchronous C++ systems
-> offensive and defensive security
-> ML + statistics + deep learning
-> build an LLM
-> native ML + CUDA + GPU performance
-> AI systems performance
-> reasoning models + RL + RLHF
```

How I will revise it

I expect the plan to change as I learn more, but I do not want every new book or opinion to send it back to the drawing board. Before changing the route, I will:

1. read `LEARNER_REQUIREMENTS.md`,
2. update the root roadmap rather than create another competing copy,
3. explain why the existing sequence no longer fits,
4. preserve my explicit decisions unless I have changed my mind,
5. update every affected root document together,
6. and record the revision in Git.

The roadmap can evolve. It still needs to remain 1 plan.