Byte Manipulation & Offset Practice
==========

1. Your self-assessment:
- You feel "trash" with byte manipulation and offsetting.
- But you’re building a proxy in Go and already dealing with buffers, truncation, and transfers.
- That’s real-world, non-toy training. You’re leveling up harder than you think.

2. Why byte manipulation matters:
- Parsing protocols (HTTP, TCP, custom binary).
- Writing state machines (tokenizers, parsers, stream decoders).
- Handling raw I/O (files, sockets, mmap, DMA, etc.).

3. Recommended Practice Resources:
A. State Machine / Byte Stream Exercises:
- Build a toy HTTP parser: read until

, extract headers, leave body in buffer.
- Write a simple tokenizer: feed characters, emit tokens (like numbers, words).
- Implement a line reader: consume arbitrary chunks but reconstruct lines.

B. Low-level Practice Problems:
- LeetCode / HackerRank: "String parsing", "Implement strStr()", "Valid parentheses".
- Advent of Code: many challenges involve parsing custom binary/text formats.
- Nand2Tetris project: parsing VM commands into bytecode.
- "Crafting Interpreters" by Bob Nystrom: tokenization + parsing.

C. System-level Projects:
- Write your own TCP stream reassembler: split and recombine packets.
- Implement Base64 encoder/decoder manually (byte shifting, padding).
- PNG file parser: validate magic header, chunk lengths, CRC.

4. Mental Model:
- Think in terms of **streams**: data is never guaranteed to be aligned.
- Maintain a **cursor/offset** while scanning.
- Use a **state variable** to track partial progress (header, body, etc.).

5. Daily Drill:
- Take any binary format (e.g., WAV, BMP, ELF, PNG).
- Open it in hex editor.
- Write a parser that extracts metadata (width, height, etc.).
- Do this often — byte streams will feel like second nature.

Remember: Strong byte manipulation = foundation of systems programming.
You're on the right path.