Post-project Writeup - Codec

James Viner

Project Summary

The task was to create a pair of command-line utilities, one to decode a specific type of packet from a .pcap file, and the other to take the textual output from the first command (or properly-formatted input in the same style) and turn it into a .pcap decodable by the first program. Additional complexity came in the form of having to convert to and from network byte order for various fields of the packet, variable-length payloads, and fields potentially containing multiple data types based on other conditions.

Challenges

Having two weeks to to a project like this that (arguably) could've been done in four or so days given the pace of previous assignments was an interesting change of pace. Having the time to sit down and research concepts subtly and not-so-subtly hinted at in the project requirements was extremely helpful, but of course, what came with it was the necessity of time-management skills and pacing myself. There was a constant temptation to do anything other than the project and it's certainly the reason I didn't start until the Thanksgiving break week (which I apologize for asking questions during). Once I actually got going, it wasn't that hard to get a bit of work done and call it a day at a reasonable stopping point, and I believe I worked at a reasonably steady pace, but there were some days where I hit a roadblock and just gave up completely.

This was also the first time I've abandoned an additional feature that I had planned on doing, that being the setting of checksum fields. My eyes glazed

over when I first read the documentation on how the checksum fields are calculated and after realizing that I already had over 15 points of features without it, I decided to pass. Given a bit more time and a greater incentive to do so, I probably could've sat down and worked through the implementation, but I also have to weigh the value of brownie points against the potential of introducing bugs to my program that are going to incidentally and catastrophically break everything, and leaving well-enough alone won out this time.

Finally, and most annoyingly, I experienced issues with my Makefile this time around. Somehow I managed to make an edit to it that didn't appear to cause any issues that stopped me from compiling, just adding in a header file that contained shared structs and functions between both of my programs. Apparently though, the particular change I had made actually was completely wrong and resulted in most of the commits in my history not being able to compile correctly. It just felt like I had a rug pulled out from under me. Make was still compiling my program perfectly well until I tried to modify my Makefile to include a second file dependency for my programs. Then, upon seeing an error I didn't fully understand, I reverted the Makefile back to the "known-good" version with git only for make to then tell me that version of the file was wrong too. I eventually figured out the problem and fixed it, but I decided to not try and fix every commit from my git history and just take the hit to my grade. I don't know if there is a moral to this one, maybe just a lesson about Murphy's law and being able to deal with it when I happen to roll snake eyes on the cosmic dice.

Successes

This is the first time I have accidentally completed a feature in one of these projects. I thought that the UTF-8 encoding feature would introduce additional complexity in some way or that there would be some sort of interaction with the packet's endianness that was going to screw things up, but I was very pleasantly surprised to see that the UTF-8 sample packet decoded properly on my first time through. It didn't occur to me until after finishing the decoder that c-style strings are arrays of single-byte chars laid contiguously in memory, so a system's endianness won't matter because it only affects data structures that take up multiple bytes per element. Pretty neat, if unintentional on my part to have it work out of the box.

I also feel like my planning paid off on this project more than on previous ones. I sat down and did a little bit of prototyping before actually starting work and laid out how I wanted certain elements of the program to look, taking into account roadblocks that I expected to hit along the way. For example, I realized that I could potentially have had a decoder that prints off line by line as it reads from the input file, but that would mean that if it came across some part of a given packet that was malformed or invalid, it would've already printed the parts it had previously decoded to the screen, and I believed it would be uglier to have an error message printed halfway through a decoded packet than to just read it all into memory and print all of the valid packets at the end. I do really feel like the effort that I put in to design the program before writing it paid off significantly more than previous projects, and I believe I can owe that to the different time constraints giving me room to breathe and think through potential

problems instead of immediately being in crunch-time mode.

Lessons Learned

If something is hinted as being helpful in the project summary, I should probably look into it. Obvious, sure, but I definitely skipped over the portion mentioning how unions would be useful in the packet and didn't revisit it until I was submitting my last few quality control commits and realized my GPS payload was printing incorrectly because of the disparate data types that can be stored in the payload. Also, doing research and reading the documentation for different functions similarly make a project significantly easier than trying to struggle-bus my way through with a surface-level understanding of the tools I'm using. For example, figuring out how htons and its sister functions work on the backend made it a lot easier to design my own similar version for 24 bit integers, rather than just copying and pasting the first 'working' solution from Stackoverflow. Similarly, I had to completely rewrite the formula for the GPS output calculations because the ones I'd ripped off of the internet were just straight up wrong. My biggest lessons really can be summed up as slowing down, planning my steps out ahead of time, and then executing once I have a good understanding of what I'll be doing and what is required to get there.