Module 8.3 - Socket Error Handling 101

Required reading materials

- [2] Lewis Van Winkle, “Hands-On Network Programming with C". Packt Publishing. May 2019. ISBN: 9781789349863. https://learning.oreilly.com/library/view/hands-on-network-programming/9781789349863/
    - Read chapter 13 - Error handling.

This module covers common error handling techniques in network programming in C.

When a networked program encounters an error or unexpected situation, the normal program flow is interrupted. This is made doubly difficult when designing a multiplexed system that handles many connections concurrently.

Event-driven programming paradigm:

Event-driven programming can provide the technique needed to simplify this logic a bit. Your program is structured so that a data structure is allocated to store information about each connection. Your program uses a main loop that checks for events, such as a readable or writable socket, and then handles those events. When structuring your program in this way, it is often easier to flag a connection as needing an action, rather than calling a function to process that action immediately.

In any case, a robust program design dictates that you carefully consider how to handle errors. Many programmers focus only on the happy path. That is, they take care to design the program flow based on the assumption that everything goes correctly. For robust programs, this is a mistake. It is equally important to consider the program flow in cases where everything goes wrong.

In real-world programming, we may want to display a text-based error message in addition to the error code.

In addition to retrieving the errno number on a Linux system, we can also get a string representation of the error by calling the strerror() function.

```C
    char *strerror(int errnum);
        The strerror() function returns a pointer to a string that describes the
error code passed in the argument errnum.

    Usage:
        return strerror(errno)
        or
        printf("%s\n", strerror(errno));
```

```

Error handling takes time and practice. My approach has been to learn how different languages and libraries handle different types of errors and apply those techniques whenever a similar situation arises. Reading the man pages and library documentation for the socket functions was also beneficial, the man page usually provides advice on how to handle the errors returned by specific function(s).

**Here are additional resources on the topic:**

https://subscription.packtpub.com/book/cloud-and-networking/9781786463999/1/ch01lvl1sec09/handling-socket-errors-gracefully

https://programmingduck.com/articles/error-catching-handling

https://www.datacamp.com/tutorial/a-complete-guide-to-socket-programming-in-python
```