Example Project : Echo Server that Serves Multiple Clients

Introduction

You are given the codes for an echo server and a client. The echo server works in a simple way: it sends back to the client anything it receives. At the moment, it serves clients one by one, in a sequential manner. This means that only one client can be served at a time, and the next client can only be served after the current client has terminated.

In this project, we **use** `epoll()` [1] **to support concurrent connections.** `epoll()` allows programs to multiplex input and output through a series of file descriptors. More detailed instructions on how to use `epoll()` can be found here.

Test Your Implementation

To test the code, you are also provided with a `checker`.py file. This python script has 6 arguments:

- `server ip:` the IP address or hostname of your server
- `port number:` the port number where the echo service is running on
- # of trials: the number of runs
- # of reads and writes per run
- max # bytes to write at a time
- # of concurrent connections

Note the following:

- The # of reads and writes per run cannot be larger than the # of concurrent connections when testing.
- There is no limit to the number of clients the server supports.

The program will output "Success!" if the implementation executes correctly. An example execution result is shown as follows:

```
mininet@vm:~/echo-server-proj$ python checker.py
localhost 9999 10 100 100000 100
```

[1] https://man7.org/linux/man-pages/man7/epoll.7.html

```
Success!
```

Useful Links

Programming in Python

The 8th edition of the textbook (and what we've discussed in class) uses sockets in Python. A Python socket tutorial is http://docs.python.org/howto/sockets.html

It may also help by reading the system implementation of epoll at https://codebrowser.dev/glibc/glibc/sysdeps/unix/sysv/linux/sys/epoll.h.html. You will better understand the epoll events.