UNIX Queen

This lab aims to practice implementing a UNIX domain socket client that interacts with a server. You must implement an N-Queen solver and upload the solver executable to the challenge server. To solve the puzzle, the solver must interact with the Queen server using a UNIX domain socket connection stored at **/queen.sock**.

The Challenge Server

The challenge server can be accessed using the nc command:

nc inp.zoolab.org 10850

You have to solve the Proof-of-Work challenge first, and then you can activate the challenge server. The challenge server has two channels to receive commands. One receives commands from the terminal, and the other receives commands from a ***stream-based UNIX domain socket*** created at the path **/queen.sock** (on the server). The commands and the corresponding responses are the same for both channels, but no welcome messages are sent via the UNIX domain socket. Therefore, you can play with the challenge server to get familiar with the commands and responses. With pwntools, you can play with our challenge server using the scripts **play.py**(view | download). You also have to download the required **solpow.py**(view | download). You can also use the play.py script to upload your solver (pass its path as the first argument to the script).

You are requested to solve the N-Queen challenge in this lab via the UNIX domain socket. The challenge server allows you to optionally upload a compressed N-Queen solver binary file encoded in base64. The uploaded solver can interact with the challenge server via the UNIX domain socket. The UNIX domain socket reads commands sent from a client to the server and sends the corresponding response to the client.

Command responses for cell number placement and puzzle checks requested from the UNIX domain socket are also displayed on the user terminal. Your solver program may also output messages to stdout or stderr for you to diagnose your solver implementation.

Your solver must be compiled with the -static option.

The commands used for the challenge server are listed below for your reference.

- H: Show this help.

- P: Print the current puzzle.

- M: Mark Qeeun on a cell. Usage: M [row] [col]

- C: Check the placement of the queens.

- S: Serialize the current board.

- Q: Quit.

Most commands do not have arguments except the M command. The M command

places a Queen in an empty cell. Once you have filled all the N queens, you must invoke the C command to perform the check. A success message is displayed if all the queens have been filled without any conflict.

You may want to use our crossbuild docker on simplify the building process on your machine, e.g., Apple Silicon (M1/M2) platform. The file can be downloaded from here. The procedures to build your program is as follows:

```
tar xjf crossbuild.tbz
(cd crossbuild; sh build.sh)      # it will create a chuang/crossbuild docker image
# switch to your working directory
# suppose your source code is placed in solver.c
docker run -it --rm -v "`pwd`:/build" --user "`id -u`:`id -g`" -w /build
chuang/crossbuild x86_64-linux-gnu-gcc -Wall solver.c -o solver -static
```

If your system does not have **sh** (e.g., Windows cmd prompt, not in WSL), you can build the image and compile your source code using the commands:

```
docker build -t chuang/crossbuild .
docker run -it --rm -v ".:/build" -w /build chuang/crossbuild x86_64-linux-gnu-gcc -Wall solver.c -o solver -static
```

Demonstration

- [10 pts] Your uploaded solver can output a message via stdout or stderr.
- [30 pts] Your uploaded solver can connect to the UNIX domain socket. The challenge server shows a message indicating that a client has been accepted.
- [30 pts] Your solver can place Queens on empty cells. Each cell is worth 2 pt, with a max of 30 pts (15 Queens).
- [30 pts] Your solver can solve the N-Queen puzzle and pass the check. You have to repeat running your solver three times, and each successful run is worth 10 pts.