Programming Project: netpong

Description

Software evolves. You write a program that meets the specifications, and a few days later, or a few months later, or years later, someone wants you to change the program. You may need to add extra features. How easily can your original design be adapted? Did you include enough comments and clear structure so you can understand what you were up to the first time?

A typical change to software these days is to make the program work on the Internet. Changing a program to work on the Internet requires an understanding of network programming and an ability to understand and modify existing code.

For this project you will make your single-user, single-machine pong game into a two-player Internet-based program. By enhancing your code this way, you will be able to work on several important ideas and skills. You will gain experience with network programming, you will learn to make sense of and write code to a network protocol, you will write client code and server code, and you will be able to review how well your original design holds up to the demands of software evolution. Additionally, you will have a chance to learn about the Unix select() system call and write a program that waits for input from two sources at once. Finally, you will discover that this netpong game contains all the ideas you need to write a network chat program. In fact, it is a network chat program.

Here are the details:

The Virtual Playing Field

The virtual playing field looks like:

```
          ---------------------------------------------------------
                                    .
                                    .
                                    .                             #
                                    .                             #
                                    .                             #
                                    .                             #
                                    .                             #
#                                   .    O                        #
#                                   .
#                                   .
#                                   .
#                                   .
#                                   .
                                    .
                                    .
                                    .
                                    .
-----------------------------------------------------------
```

This pong screen has two parts, the left half of the court and the right half of the court. A net separates the left half of the court from the right half of the table. The ball moves just as it did before, bouncing off the walls and the paddles. What happens, though, when the ball gets to the net?

The Split Court Model

The game has an interesting twist, though. Each player only sees his or her half of the playing field. The two players see these two images:

```
         PLAYER ONE'S VIEW      PLAYER TWO'S VIEW
       ----------------------    ----------------------
                            . .
                            . .
                            . .                           #
                            . .                           #
                            . .                           #
                            . .                           #
                            . .                           #
   #                        . .       O                   #
   #                        . .
   #                        . .
   #                        . .
   #                        . .
   #                        . .
                            . .
                            . .
                            . .
                            . .
          ------------------  ----------------------
```

Player one sees the left half of the field, and player two sees the right half of the field. That is, each player sees only his or her court and cannot see what is happening on the other half of the field.

The two players can be anywhere on the Internet, each playing half the game against an unseen opponent. When the ball reaches the net, it vanishes from that court and appears through the net on the other player's screen. That player needs to bounce it back. If that player misses the ball, the opponent hears the news that the ball went out of play.

If your opponent returns your shot, the ball will reappear at some time through the wall.

Do you have the picture now?

The Project: User View of netpong

Write one or two programs that do the following:

[A] One player starts up a game as a server listening at a port he or she specifies on the command line. For example: **netpong 2001** will start netpong in server mode on port 2001. That program reports that it is waiting for a connection.

[B] The other player starts up a game as a client and tells the game to connect to the server's hostname and port. For example: **netpong xyz.myschool.edu 2001** start a client and instructs it to connect to a port on a machine.

[C] Once a connection is established, the client serves, and the server waits for the ball to appear on his or her screen. The screen for netpong is slightly different from the screen for the single-player version of pong. Somewhere on the screen netpong displays your score, your opponent's score, and any messages that are sent as part of the communication between the two programs.

[D] The client program serves the first ball. The server program waits for that ball. When the ball is in your court, the game looks and works just like regular pong. When the ball reaches the `net' it does not bounce. Instead, it vanishes from your court and appears on your opponent’s court with the same row, speed, and direction it had when it left your court.

[E] When the ball is in the other person's court, the game looks and works just like regular pong. When the ball reaches the `net' it does not bounce. Instead, it vanishes from the other court and appears on your court with the same row, speed, and direction it had when it left the other court.

```
------------------------------------------------------------
.
.                                                                       #
.                                                                       #
.                                                                       #
.                                                                       #
.                                                                       #
.                                                                       #
.
.
.
.
.
.
.
.
.
------------------------------------------------------------
Me: 0  smith: 0              Waiting for serve..
```

[F] If a player misses a ball. The other player gets a point, and the other player gets the next serve. If there are no more balls left, the game is over, and both sides end the game.

[G] The server program goes back to waiting mode, ready to accept another connection to play another game.

[H] Either player may press the Q key to quit the game. In that case, the other player is notified and the game ends. If the ball is not in a player's court, the Q key may not register immediately. For extra credit, you may make the quit and paddle motion actions available all the time.

How it Works: SPPBTP

There are two processes running, one for each player. Those processes need to communicate. When the ball leaves one court, that process has to tell the other process the position, speed, and direction of the ball. When a player misses a ball, his or her process has to tell the other process so the other player can serve. When the last ball goes out of play, the two processes need to end the game and allow the players to exchange any congratulatory remarks. How do two processes communicate? Many Internet programs use a TCP connection the way people use a telephone connection. One calls the other, and when the connection is established, they send messages and responses back and forth using an agreed-upon protocol. For this assignment, you will use the simple ping pong ball transfer protocol, known as SPPBTP. This protocol is described in a document called RFCNP. A copy of the document is available in the assignment folder. You may build the project as two separate programs (a server version and a client version) or you may build the project as one combo program (able to be a server or client depending on command line options.) Regardless of whether you write two programs or one, most of the code will be the same.

Instructions

There are many ways to approach the project of enhancing software. One way is to first think about what changes you need to make to your existing program. Your pong program probably works ok. If you think about it carefully, you may be able to add these new features without ripping the original into pieces.

Phase I (1-4)

[1] How hard is it to change the code so the game can appear with the paddle on the left or on the right? Can you replace the constants for paddle and wall positions with variables?

[2] Look through your pong code and review how the program works. Look at the part where the ball bounces off the far wall. That part will need to be changed. In the new version, the ball will not bounce but will instead be passed through the socket to the other process.

[3] Look at the part where the ball misses the paddle. In the original game, missing the paddle caused the game to serve a new ball. In the new version, the other player serves the new ball.

[4] Think about the timer. When the ball is on the other court, the ball moves under the control of a different ticker. Do you need to keep your ticker running while the ball is in the other process?

Phase II (5-8)

[5] What about the paddle? When the ball is in the other process, do you want to allow your player to move the paddle?

[6] What does your program do when the ball is in your court? What does the program do when the ball is in the other court? How does your program make this transition?

[7] How does the ball travel from court to court? What does the other process need to know about the ball? What do you need to know from the other process when it sends the ball back?

[8] How do you keep score? Now read the RFC to see how the SPPBTP helps make sense of the problems raised by the preceding questions. This program is different from some standard Internet client-server pairs. This program starts off as a clear client-server pair, but then settles into a pair of symmetric programs as they pass the ball back and forth. Both sides can send a ball over the net, and both sides can miss the ball.

Phase III

Three Sections: Start, Middle, and End

The game has three parts. At the start, the server accepts a call from the client, and they exchange names and operating parameters. The server invites the visitor to serve. That ends the start of the game. The middle part is an alternation of pong games. One process runs the regular pong code and the other side waits for the ball to return. When the ball reaches the net, the two processes change roles. What else happens in the middle part of the game? The game ends when one player misses the last ball. Read the RFC to see how that closing exchange is defined.

States and Errors, Testing

Internet client-server programs are often described in terms of operating in various states. The game is in the `waiting' state when it waits for the ball to arrive. When the ball arrives, it changes to the `play' state. If a QUIT message appears, though, it changes to the finished state. It will help you a lot if you draw out a chart showing the various states the program can be in and how the program responds for each message it receives over the network or for each event that occurs as the ball moves around its court. If the game is in the waiting state and receives a HELO message, something is wrong. Some messages are not appropriate in a certain state. The simplest thing to do is to send an error message to the other process, tell your user there has been an internal error, and then exit. You could think about more sophisticated error handling and recovery, but coding it is not part of the assignment.

Testing your program is not all that tricky. Since the protocol is plain text, you can start up a server and use telnet to talk to the server in plain ascii.

Extra Credit - select () or poll ()

When the ball is off the court, the process waits for a message from the other process. What if the user wants to move the paddle up and down while waiting for the ball to appear? What if the player presses Q while the game is waiting? For ten points of extra credit, read about and use the select () or the poll () system call to allow your program to watch for input from the keyboard as well as from the socket. By doing so, you can allow the user to move the paddle even when the ball is in the other court, and you will allow the user to press Q when the ball is in the other court and have the other player hear about it at once.

Submission

Phase I

Answer questions 1-4 above (p.4) with all algorithms and charts.

Phase II

Answer questions 5-8 above (p.4) with all algorithms and charts.

Phase III

Complete the three sections of the game and implement them in C. Submit (a) your source code, (b) a Makefile, and (c) a run of a clean compile. This is curses-based program, so a script will not look good. We'll run the program.

A Picture

Here is some ASCII art depicting the various components of the netpong system:

```
         +---------------+                 +-----------------+
         |      +------+ |                 |  +------+       |
   crt   |      |timer|  |                 |  |timer|        |   crt
 +-----+ |      +------+ |                 |  +------+       |  +-----+
 |    |==|               |                 |                 |==|     |
 +-----+ |  IN_PLAY      |                 | WAITING         |  +-----+
======== |  responds to  |                 | responds to     | ========
  kbd    |  kbd, timer   |                 | socket          |   kbd
         |  (opt: socket)|                 | (opt: keyboard) |
         +-----| |-------+    Internet     +---| |-----------+
               | |_____________________________| |
               |_________________________________|
```

Future Directions

By writing your netpong client and/or server to fit the specifications of the SPPBTP, your client should be able to play pong with any server, and your server should be able to accept games from any SPPBTP-compliant client.

In fact, you could write programs to simulate another player. A pong-bot could accept a ball and then fire it back at lightning speed, or it could hold onto the ball for a long time until it thinks you have lost interest and then lob one back.

How about writing proxy programs to allow people to play doubles, that is, with two people on each side. If you have the time and interest, see if you can figure out a system for this.

What would it take to be able to issue invitations to users on other machines to play netpong with you? Finally, can you use the ideas from this program to create a chat program, like the Unix talk program?