LAB 8

CS 361: Systems Programming / Spring 2023

Description

In this lab session, you will:

- Set up a server process
- Explore function `fstat`

Please read this document carefully and follow the instructions on the last section to complete this lab session. When you answered all the questions, please show your work to the TA.

Guide

1. Accept the invitation for Lab 8 on Github classroom: https://classroom.github.com/a/CMIA-YTO
2. Import the Github repository created to your machine using vscode, as explained in Assignment 0
3. Make sure that you can launch a terminal inside vscode via menus: Terminal > New Terminal
4. Read this guide and answer the questions as they appear. You should answer a total of 11 questions.

Running two processes: client and server

In this lab, you will use the command `make` to compile all programs, just as in earlier labs. You will need to use two terminals: one for the server process and one for the client process.

Port

For this assignment, you will need to pick a port to use. This is a 5-digit integer. If you pick the same port as another student, your assignments will not work. You can use the last 5 digits of your UIN, or go to www.random.org to pick a unique port number.

Program lab8-1 and mystery take the port number you picked as argument (e.g., ./mystery <PORT>)

Structure of a server in C

Lecture 14 has information about how to create a server in C. Open the slides for that lecture so you can add all the code you need.

Socket

The first step is to create a socket. This is a file descriptor created with the function socket:

```
int socket(int domain, int type, int protocol);
```

Question 1: Refer to the slides of Lecture 14 and add code to lab8-1.c to create a socket for IPv4 and TCP. The socket call should return a positive number. What is the number that socket returns?

Bind

The second step is to bind a socket to an address and a port:

```
int bind(int sockfd, struct sockaddr *addr,  socklen_t addrlen);
```

Question 2: Refer to the slides of Lecture 14 and add code to lab8-1.c to bind the socket you created to address 127.0.0.1 and the port you passed as argument. How did you initialized the 2nd argument addr so that the call to bind returns 0?

Hint: You can use function perror to print error messages when bind returns non-zero.

Listen

The third step is to listen a socket:

```
int listen(int sockfd, int backlog)
```

Question 3: Refer to the slides of Lecture 14 and add code to lab8-1.c to listen the socket. What value did you choose as the backlog? Make sure that the call to listen returns 0.

Accept

The final step is to accept incoming connections on the server socket:

```
int accept(int sockfd, struct sockaddr * addr, socklen_t * addrlen);
```

Question 4: Refer to the slides of Lecture 14 and add code to lab8-1.c to accept an incoming client. Run lab8-1 in one terminal, and mystery in another. What is the number of the socket that accept returns?

Reusing addresses

When a server crashes before receiving all the data from the client, the operating system prevents new processes from binding to the same port for a short while (e.g., one minute). You can disable this behavior with the following code snippet, between socket and bind:

```
int optval = 1;
setsockopt(<server socket>, SOL_SOCKET, SO_REUSEADDR, &optval, sizeof(optval));
```

Question 5: Run lab8-1 as described in question 4 two times in a row. Can you bind to the socket in the second try? Does this change after adapting the snippet above to your code?

Note: Answers to question on Piazza about this snippet will be endorsed only if they do not include the code (e.g., Q: "Why is my server not able to bind a socket after it crashes?" A: "Please check Lab 8 about reusing addresses").

Receiving data

Function `recv` allows you to receive data through a socket:

```
ssize_t recv(int sockfd, void *buf, size_t len, int flags);
```

Question 6: Refer to the slides of Lecture 14 and add code to lab8-1.c to read data from an accepted client. Pick a reasonable buffer size. What is the message that program mystery sends to your server?

Question 7: Program mystery sends a second message. What code allows you to receive the whole message? How can you print the message sent by program mystery?

- Note 1: the message may not be NULL terminated, so functions like printf may not work as you expect. Refer to Lab 4 about functions that write a given amount of data to file descriptor 1.
- Note 2: the message sent ends in a new line. Make sure that you print the whole message, including the new line which is the last character.

Sending data

Function `send` allows you to send data through a socket:

```
ssize_t send(int sockfd, const void *buf, size_t len, int flags);
```

Question 8: Refer to the slides of Lecture 14 and add code to lab8-1.c to send a message to the client. What does the client print when it receives your message?

Question 9: How can you send a message that the client simply prints back?

File meta-data

Function fstat allows you to read metadata about a file:

```
int fstat(int fildes, struct stat *buf);
```

This function takes a pointer to a `struct stat` (or the address of a `struct stat` on the heap/stack), and fills it with information about an open file-descriptor `fildes`. Struct stat has the following fields:

```
struct stat {
          dev_t     st_dev;         /* ID of device containing file */
          ino_t     st_ino;         /* Inode number */
          mode_t    st_mode;        /* File type and mode */
          nlink_t   st_nlink;       /* Number of hard links */
          uid_t     st_uid;         /* User ID of owner */
          gid_t     st_gid;         /* Group ID of owner */
          dev_t     st_rdev;        /* Device ID (if special file) */
          off_t     st_size;        /* Total size, in bytes */
}
```

Question 10: Read the description about `struct stat` above. Change file lab8-2 to open a file descriptor to file Makefile and populate a `struct stat` with information from function `fstat`. What code do you need to print the size of the file?

Question 11: Use command `ls -l` on the terminal to discover the size, in bytes, of file Makefile. Does it show the same size that you answered for Question 10? If not, then you should change Question 10 so that it shows the same size for file Makefile as `ls -l`.

Extra / Optional

- Can you write your own client that connects to lab8-1, instead of using program mystery?

Grading

Show your UIC card to the TA when you enter the lab, or type your UIN on the chat when joining remotely. Stay in the session until you show your work, or until the TA announces that the lab is over.

- You have to remain present for the whole lab to get attendance, which you can then use to resubmit Assignment 4.
- You can leave early after showing your work to the TA (answers to all questions). In this case, you will get a 5% bonus in Assignment 4.