CPSC 3600 Homework2
Spring 2023 (50 points)

Creating a Simple File Server with TCP sockets

Instructions:
This assignment is done individually. You *cannot discuss the solution to this programming exercise* with anyone except your teacher or TA. Please review TCP code in Donahoo.

Exercise
In this exercise you will create a Simple File Server. Your server will provide the following services to the users.

1. Server will keep track of user names, their IP addresses and what files they have downloaded. The first time user connects to the server, he will be asked to create a username.

Server will then store this username along with user's IP address in their "database" (you can devise some clever way of doing this). This user database can be a file that contains users and their IP addresses, or some dynamically created entity. Every time the user downloads a file, database will be updated to keep a record of it. When you are done testing, the user database will need to be deleted. (by *make clean* command?)

2. User can request a list of all files available for download. The Server will send the user a list of files that display file name and size for each file.

3. User can download a file from the server. (User will have to list all files first to select what they want.) This downloaded file will be saved in the user's current directory with original formatting, and will also be displayed on the screen.

Files to Create:
You will create two files, server file named *server.c*, and client file named *client.c*. You will also need several small text files, so your server has some files for the user to download. You will need to make a simple *makefile* with *make* and *make clean* targets that also removes the user database. Make will produce executables named *fileserver* and *client.* Submit a tarball with these 3 files, tarball should contains no subdirectories.

It is up to you to figure out the details. There are several different solutions to this problem, and everyone's solution may be different. I do recommend that you review how to read/write files in C and go over the code line by line following the Chapter 2 in Donahoo.

To run your executable you will login to three different SoC machines (one server and two clients) and figure out their IP addresses (you can use *ifconfig*). Tile the three terminals on your screen so you can see all three at the same time. Start server first.

Example output of the communication

Server

$ ./fileserver <port>

```
Welcome to Simple File Server

Connected to:    johnny, IP: xxx.xxx.xx.xx
    johnny requested file listing
    johnny requested download: file1.txt
Sent file1.txt to johnny
Connection with johnny terminated

Connected to:        kevin, IP: yyy.yy.yy.yyy
    kevin requested file listing
    kevin requested download: file2.txt
Sent file2.txt to kevin
    kevin requested listing of downloads
Sent listing of downloads to kevin
Connection with kevin terminated
```

Client A:

$ ./client <Server IP address> <port>

```
Select an action from the menu:
1. Request file listing
2. Download file
3. List all downloads
4. Exit

>> 1
>> Please enter your username
>> johnny
>> Listing received from server:
filename      size
file1.txt     256 bytes
file2.txt      12 bytes
asg1.c         56 bytes

Select an action from the menu:
1. Request file listing
2. Download file
3. List all downloads
4. Exit

>> 2
>> Enter filename you would like to download:
>> file1.txt
>> Received file1.txt
>> Displaying file1.txt:

.....File 1 contents displayed here...
(file1 should also appear in the current directory)


Select an action from the menu:
1. Request file listing
2. Download file
3. List all downloads
4. Exit

>> 4
>> Goodbye!!!
```

Computer B:
$ ./client <Server IP address> <port>

```
Select an action from the menu:
1. Request file listing
2. Download file
3. List all downloads
4. Exit

>> 1
>> Please enter your username
>> kevin
>> Listing received from server:
filename      size
file1.txt     256 bytes
file2.txt      12 bytes
asg1.c         56 bytes

Select an action from the menu:
1. Request file listing
2. Download file
3. List all downloads
4. Exit

>> 2
>> Enter filename you would like to download:
>> file2.txt

>> Received file2.txt
>> Displaying file2.txt:
```

```
.....File 2 contents displayed here...
(as above, file should appear in the current directory)

Select an action from the menu:
1. Request file listing
2. Download file
3. List all downloads
4. Exit

>> 3
>> Listing all downloads
johnny        file1.txt        256 bytes
kevin         file2.txt         12 bytes

Select an action from the menu:
1. Request file listing
2. Download file
3. List all downloads
4. Exit

>>4
Goodbye!!!
```

What to Research:

1. Research the *dirent* structure. (directory entry) Include <dirent.h> header file. This can be used by the server to obtain the file listing. Do not hard code filenames.

```
struct dirent {
       ino_t          d_ino;              /* inode number */
       off_t          d_off;              /* offset to the next dirent */
       unsigned short d_reclen;           /* length of this record */
       unsigned char  d_type;             /* type of file; not supported
                                             by all file system types */
       char           d_name[256];        /* filename */
};
```

You can study sample code located in Files on Canvas to list directory contents. Please note that file sizes are not listed in the sample code. *You will need to figure out a way to accomplish that.*

2. This is not a difficult assignment, and you have the starter code to help you out. Work in incremental manner. To check if file really did get downloaded, do a directory listing (ls) in another terminal. Open file to make sure it has its original formatting.