CSCI 4220 Assignment 1

TFTP Server

Due Date: Wednesday, September 25, 11:59:59 PM

Your task for this team-based (max. 2) assignment is to implement a TFTP server according to RFC 1350 using C or C++.

Your server should be able to support multiple connections at the same time by calling the `fork()` system call which you may be familiar with from Operating Systems. We will discuss it on September 11th. You MUST support the "octet" mode. You should not implement the "mail" mode or the "netascii" mode.

Upon not receiving data for 1 second, your sender should retransmit its last packet. Similarly, if you have not heard from the other party for 10 seconds, you should abort the connection.

Take care to not allow the Sorcerer's Apprentice Syndrome (SAS). Don't worry, the implementation in the RFC has already corrected SAS. Additionally, we will only be testing files smaller than 32 MB.

Be sure when testing to set the mode to binary. You might find it useful to use an existing tftp client - if you are in Ubuntu/WSL one is available through `sudo apt get install tftp`

`SIGALRM` is discussed in our textbook (Chapter 14) which may be useful when implementing timeouts. We will discuss signals on September 16th.

As we will not be able to use root privileges on Submitty, you should NOT be requesting port 69 (which is a reserved port). Your program will instead take 2 integer arguments, `[start of port range] [end of port range]`. Your server should start by listening on the first port in that range (this will be used instead of port 69), and instead of using a random port for TIDs should use the next highest port in the range. For simplicity you can assume that during execution of your server, once a port has been used in a TID it cannot be used for a new TID.

Please include a README.txt file which should include your name, the name of your partner, and any helpful remarks for the grader. Also submit a Makefile - it can be quite simple and there are many online references, in addition to the examples in the book's code. You can test it by running `make`. It must generate an executable called `tftp.out` Remember that file names are case sensitive on Submitty!

Please make sure to get started early!

Regarding Makefiles, I've borrowed the following text from Netprog Spring 2018:

```
I realize I required that everyone submit a Makefile so that negates the cpp requirement.
That said, if you want to use cpp, you are free to do so.  One trick with doing that, though,
is that the unp.h header (and matching library) is written entirely in C.  If you wish to use
that in your cpp file, you need to do the following:

extern "C" {
#include    "unp.h"
}

I'm not sure if the majority of the class is familiar with this or not but basically it prevents
C++ name mangling for all of the functions declared inside of the unp.h header.
You can read more details here:
```

https://isocpp.org/wiki/faq/mixing-c-and-cpp