ECE/CS 438: Communication Networks **Fall 2021**

Machine Problem 0

Handed Out: August 23^{th}*,* 2021 *Due: Never*

TA: Yu-Lin Wei

Abstract

This machine problem introduces you to socket programming in C and the mechanism for submitting assignments. We use virtual machines to simulate multiple network entities. This assignment will help you prepare your environment so that all subsequent assignments will be simpler to code and submit. Since this assignment has all the preparatory material, the TAs will not help you understand this part in subsequent machine problems.

1 Introduction

The purpose of this machine problem is to familiarize you with network programming in the environment to be used in the class and to acquaint you with the procedure for handing in machine problems. The problem is intended as an introductory exercise to test your background in C programming. You will obtain, compile, run, and extend a simple network program. The extensions to the code will introduce you to one method of framing data into individual messages when using a byte stream abstraction such as TCP for communication.

2 What Is Expected in this MP?

Inside the release folder, You will find a folder named mp0, which contains the programs `client.c, server.c, talker.c,` and `listener.c` — all from Beej's Guide to Network Programming:

`(http://beej.us/guide/bgnet/)`.

Beej's guide is an excellent introduction to socket programming, and very approachable. Compile the files using `gcc` to create the executable files `client, server, talker,` and `listener` . We provide a Makefile that will compile all 4 (simply run `make` inside the directory). The real assignments will require you to submit a Makefile, so if you aren't already experienced with make, please familiarize yourself with the provided Makefile, and ensure that you can adapt it to a new project. Login to two different machines (Virtual Machines), and execute `client` on one and `server` on the other. This makes a TCP connection. Next, execute `talker` on one machine and `listener` on the other. This sends a UDP packet.

Note that the connection oriented pair, `server` and `client`, use a different port than the datagram oriented pair, `listener` and `talker`. Try using the same port for each pair, and run the pairs simultaneously. Do the pairs of programs interfere with each other?

Next, change `server.c` to accept a file name as a command line argument and to deliver the length and contents of the file to each client. Assume that the file contains no more than 100 bytes of data. Send the length of the file (an integer between 0 and 100) as an 8-bit integer. Change `client.c` to read first the length, then that number of bytes from the TCP socket, and then print what was received.
The client output should look like this:

```
client:  connecting to <hostname>
client:  received <filelen> bytes
This is a sample file that is sent over a TCP connection.
```

where <hostname> is the address of the server, <filelen> is the number of bytes received, and the rest of the output is the file contents. That's it. Sounds simple, doesn't it? Indeed, for experienced Unix/C programmers this MP is trivial. Others should find it a nice way to get started on network programming.

You will need to have (or quickly acquire) a good knowledge of the ANSI C programming language, including the use of pointers, structures, typedef, and header files. If you have taken CS241 you should already have the necessary background. Don't simply download the source code and compile the programs, but make sure that you read and understand how the sockets are created and the connection established. Beej's guide is a very useful tool in this sense.

3 How to Set Up VirtualBox VM Environment?

The autograder runs your code in VMs—64-bit Ubuntu 16.04.3 LTS VMs (desktop version), running on VirtualBox. Therefore, to test your code, you will need a 64-bit Ubuntu 16.04.3 LTS VM of your own. (Even if you're already running Ubuntu 16.04.3 LTS on your personal machine, later assignments will use multiple VMs, so you might as well start using the VM now.)

WARNING: COMPILATION CAN BE A LOT LESS PORTABLE THAN YOU THINK, ESPECIALLY WHEN OSX OR EWS IS INVOLVED. Please don't assume that it will be ok after testing it only on your personal machine or EWS. (Just don't use EWS at all; it is not well suited to classes that involve networked programming assignments.)

A tutorial for installing Ubuntu on VirtualBox can be found at

`http://www.psychocats.net/ubuntu/virtualbox`.

This tutorial is for Windows, but VirtualBox works and looks the same on all OSes. The Ubuntu 16.04.3 image:

`http://www.ubuntu.com/download/alternative-downloads`

After the Ubuntu install process (within the VM), you should install the ssh server. You can do `sudo apt-get install openssh-server` once the OS is installed. Use `apt-get` (`sudo apt-get`

`install xyz`) to install any programs you'll need, like `gcc, make, gdb, valgrind`. I would suggest also getting `iperf` and `tcpdump`, which will be useful later.

4 How to Set Up Networking Inside VMs?

VirtualBox's default network setup is a NAT (which we'll learn about later!) interface to the outside world, provided by the host computer. This allows the VM to access the Internet, but the host computer and other VMs will not be able to talk to it. We're going to replace the NAT interface with one that allows those communications.

However, BEFORE YOU MAKE THIS CHANGE, you should use that Internet access to: `sudo apt-get install gcc g++ make gdb iperf tcpdump wget`

Now it's time to replace the network interface. Make sure the VM is fully shut down, and go to its Settings → Network section. Switch the Adapter Type from NAT to "host-only", and click ok. Restart, and the VM will now be able to talk to other host-only VMs on the same computer, as well as the host computer (for `ssh`). Finally, `sshfs` is an excellent way to access the VM's filesystem (or any other remote filesystem). On your host system, `sshfs 192.168.56.101: directory-to-mount-in` will mount the VM's filesystem as if it were a USB flash drive. (Replacing the IP address with whatever it actually is, of course). `sshfs` may not be available on all operating systems.

5 What Tips and Tricks Will be Useful?

Copy-pasting directly from pdf files is a bad idea. Dashes, quotes, and kerned characters may get completely mis-represented when pasted in a terminal.

Never add compiled executables and object files to the git. You might lose points in subsequent assignments if you add executables or object files to git. Use `git add` to add only the source files (`src` folder), `Makefile`, `version.txt`, and `teamname.txt` files.

MP0 is ungraded, but still very important to get you started. Performing this assignment successfully will make submitting the subsequent assignments much easier.

6 Git Instructions

6.1 Course Setup (only once for the entire semester)

The first time you're accessing the ECE438 repository this semester, you will need to have a ECE438 repository set up for you. This process is simple:

1. Visit `https://edu.cs.illinois.edu/create-ghe-repo/ece438-fa21/`
2. The web service will generate your ECE 438 repository and provide you with your repository name. It's usually `https://github-dev.cs.illinois.edu/ece438-fa21/yourNetID`.

6.2 Workspace Setup (necessary only once per computer/directory)

Cloning your repository:

To setup your computer to work on an MP or a lab, you will need to clone your repository onto your comptuer. The URL of your repository will be based on your NetID and you will need to replace NETID with your NetID. To clone your repository, run git clone inside your desired directory:

```
git clone https://github-dev.cs.illinois.edu/ece438-fa21/yourNetID.git folderName
```

you can replace `folderName` with whatever folder you want created (for example `ece438`). You can submit this MP and all subsequent ones through this repository.

Username and password entry:

Your username is your NetID. On some systems, git (and other command line programs) will not display anything when you type your password. This is expected: type in your password as normal, and then hit Enter ⏎.

Finally, move into the directory you just cloned:

```
cd folderName
```

Released code:

The code we release as a starting point for any MP will be present in the release repository. You need to add this repository as a remote:

```
git remote add release https://github-dev.cs.illinois.edu/ece438-fa21/_release.git
```

You're now all set to begin to work on your assignment!

6.3 Assignment Setup (necessary only once per assignment)

To retrieve the latest assignments for ECE 438, you need to fetch and merge the release repository into your repository. This can be done with the following commands:

```
git pull
```

```
git fetch release
```

```
git merge release/master -m "Merging release repository"
```

6.4 Assignment Submission (do this often!)

The first step in assignment submission is to increment the version number in your `version.txt` file. Your code will not be picked up by the autograder if you fail to increment the version number.

Every time you want to submit your work, you will need to `add`, `commit`, and `push` your work to your git repository. This can always be done using the following commands on a command line while within your ECE 438 directory:

```
git add -u
```

`git commit -m "your commit message"`

`git push origin master`

You can also check the working tree status of your repository by running `git status` after each step to make sure everything has been added, commited and pushed. Be sure not to skip `origin master` when trying to `push`/`pull`.

6.5 Verifying Submission

You can always verify your submission by visiting `https://github-dev.cs.illinois.edu/` and viewing the files in your repository. Only the files that appear on your github-dev repository will be graded.

6.6 How to See Your Grade?

The autograder runs periodically on all new submissions (again, don't forget to update the version number!). The results are updated in a different branch (_grades) inside your directory.

For your convenience, we have created a script to see the results: `./see_results.sh`

The script swaps the branch to _grades, shows the results, and swaps the branch back. If you run the see_results.sh file on mp0 before the autograder has run, your directory will move to the _grades branch. You will have to manually execute `git checkout master` to get back to your working branch. DO NOT work on the _grades branch!

Tests generally take 1-4 minutes, and there may be a queue of students. You can see where you are in the queue at

`http://fa21-cs438-01.cs.illinois.edu:8080/queue/queue.html`. This is a UIUC-private IP. If your device is not accessing it through campus network, please use Illinois VPN to get a private IP.

Caution: During the hours leading up to the submission deadline the queues could be multiple hours long. So it is advisable to get your work done early.