# Machine Problem 0

***Handed Out: September 22$^{th}$, 2023*** ***Due: Never***

**Abstract**

This machine problem introduces you to socket programming in C and the mechanism for submitting assignments. e 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 the 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. 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 22.04.1 LTS VMs (desktop version), running onVirtualBox. Therefore, to test your code, you will need a 64-bit Ubuntu 22.04.1 LTS VM of your own.(Even if you're already running Ubuntu 22.04.1 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/ EWS/ WSL IS INVOLVED. Please don't assume that it will be ok after testing 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 OSesNote: If your machine is on ARM-architecture (e.g.. M Macbook), you might want to use other virtual machine solution like Docker, since VirtualBox does not support ARM architecture. Please contact usif you had difficulties setting up another virtual environment. The Ubuntu 22.04.1 image:

**https://ubuntu.com/download**

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 installxyz**) 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.

Note: WSL (Windows-Subsystem for Linux) might work as one of your machine, but keep in mind that some compiler actions might be different. If you are using WSL, please be very careful about the undefined coding behaviors, e.g., variable not initialized, memory leakage, etc.

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

# 6 Assignment Submission (MP0 does not need this step)

We use the autograder to grade your MPs, the submission process is simple.
First, open the autograder web page:

**http://10.105.100.204**.

This is a ZJUI-private IP. If your device is not accessing it through the campus network, please use VPN to get a private IP

You will see two sections. **MP Submission** allows you to submit your assignment. You can do this by entering your Student ID (start with 320), selecting which MP you are submitting, selecting the file extension of your files and uploading your MP files. **Note**: only C/Cpp files are accepted. When uploading files, **add your files one by one, do not choose multiple files to add at one time**. **Submission History** section allows you to check your submission history and grade. You can do this by entering your student ID.

Caution: The queue can only handle 200 submissions at one time, so remember to check your submission status in **Submission History** after you submit your assignment. During the hours leading up to the submission, the queue could be long. So it is advisable to get your work done early.