

# Programming Assignment 1: **Manual**

Brandon Dotson  
CS 550-01  
9/30/2017

## **Table of Contents**

|                      |   |
|----------------------|---|
| Overview.....        | 3 |
| Getting Started..... | 4 |
| Command Listing..... | 7 |

# **Overview**

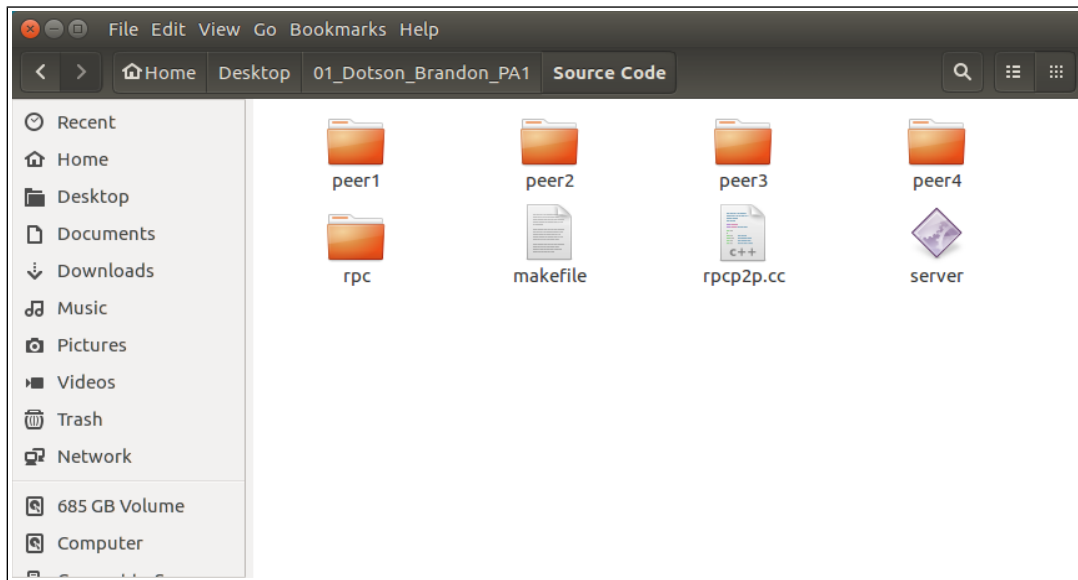
Programming Assignment 1 tasked students with creating a basic peer-to-peer file sharing program using programming languages and network implementations of our choosing. I decided to create my project using C++ and a remote procedure call-based implementation for the network. The server and peers share the same source code, and when starting, the user is prompted to select if the process will be used for the server or for peers. In order for the file sharing system to function correctly, one “server” process and at least one “peer” process is required, however more than just one peer is recommended.

Entering a value of “1” makes the process behave as the server. When the server process starts, it selects a port to listen on and binds all its relevant functions for future remote procedure calls. After this, its creation is acknowledged and it waits for peers to connect. If no peers connect, it continues waiting until it is interrupted by the SIGSTOP (Ctrl + C) signal.

A value of “2” makes the process act as a peer. When beginning as a peer, an IP address and port number are required to connect to the server. Once this is complete, the peer is prompted to enter commands which will be sent to the server for processing. It should be noted that the peers are all listening on a second thread, waiting for the eventual peer-to-peer connection. The specific commands and their uses can be found at the end of this manual.

# Getting Started

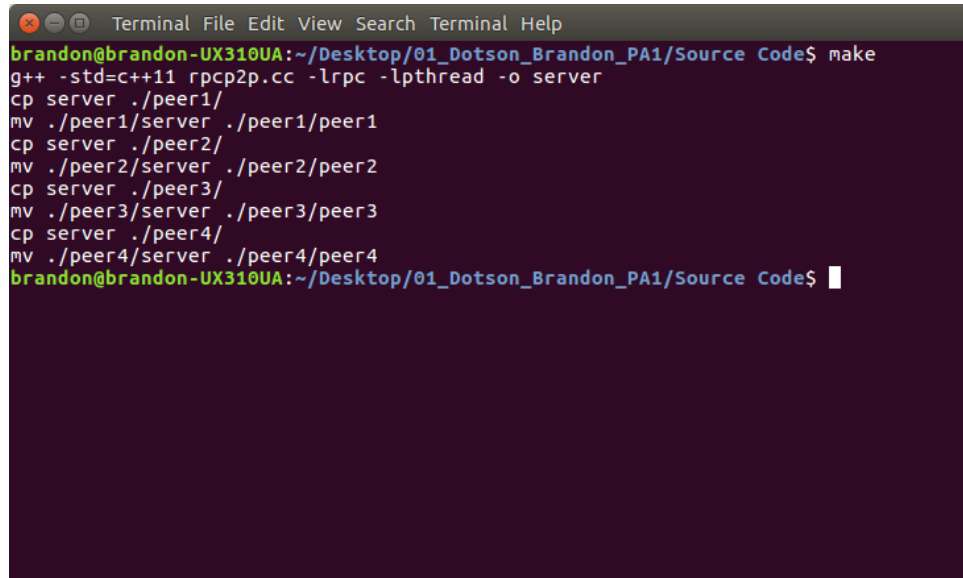
- 1) Before anything else is done, the makefile must be copied into the “Source Code” directory. For your convenience, a copy of the file will already be present.



**Figure 1: The source code directory**

- 2) Open the terminal and navigate to the “Source Code” directory.
- 3) Enter the following command: make (You may use make clean; make if you would like to do it a second time without the hassle of deleting files).
- 4) If the make command fails due to a fatal error involving “rpc/client.h”, do the following:  
navigate to the rpc/rplib directory. Delete the build folder. Type the following commands:  
  
“cmake ..” (after installing cmake)  
  
“cmake --build .”  
  
“sudo make install”

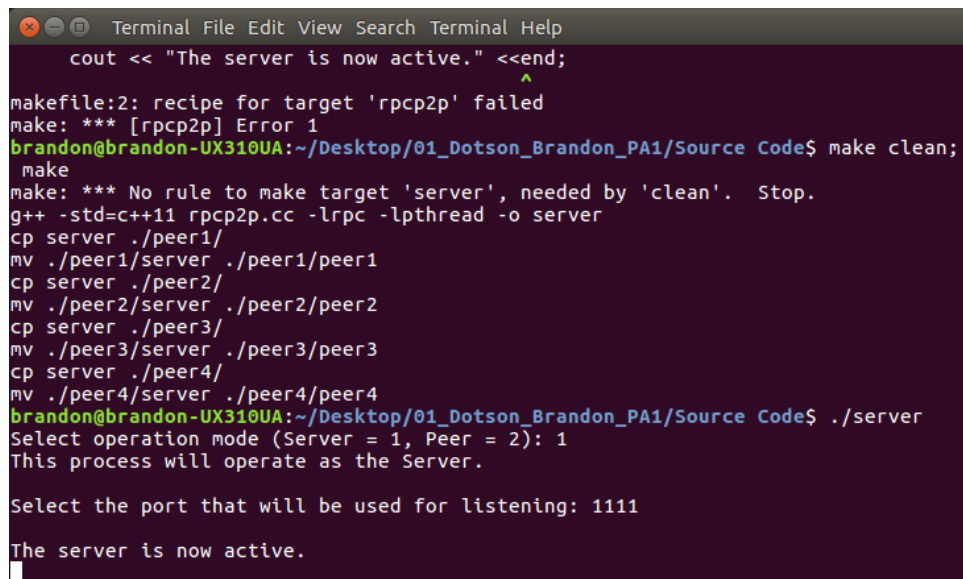
- 5) The supplied makefiles should work. Go back to step 3.



```
Terminal File Edit View Search Terminal Help
brandon@brandon-UX310UA:~/Desktop/01_Dotson_Brandon_PA1/Source Code$ make
g++ -std=c++11 rpcp2p.cc -lrpc -lpthread -o server
cp server ./peer1/
mv ./peer1/server ./peer1/peer1
cp server ./peer2/
mv ./peer2/server ./peer2/peer2
cp server ./peer3/
mv ./peer3/server ./peer3/peer3
cp server ./peer4/
mv ./peer4/server ./peer4/peer4
brandon@brandon-UX310UA:~/Desktop/01_Dotson_Brandon_PA1/Source Code$
```

Figure 2: Using the makefile

- 6) Open the “server” program file. Select whichever port you prefer (**Note:** Do not use ports 9000, 9001, 9002, ... 9010, etc. These will be used by the peers. Please make sure these are available).



```
Terminal File Edit View Search Terminal Help
cout << "The server is now active." <<end;
^
makefile:2: recipe for target 'rpcp2p' failed
make: *** [rpcp2p] Error 1
brandon@brandon-UX310UA:~/Desktop/01_Dotson_Brandon_PA1/Source Code$ make clean;
make
make: *** No rule to make target 'server', needed by 'clean'. Stop.
g++ -std=c++11 rpcp2p.cc -lrpc -lpthread -o server
cp server ./peer1/
mv ./peer1/server ./peer1/peer1
cp server ./peer2/
mv ./peer2/server ./peer2/peer2
cp server ./peer3/
mv ./peer3/server ./peer3/peer3
cp server ./peer4/
mv ./peer4/server ./peer4/peer4
brandon@brandon-UX310UA:~/Desktop/01_Dotson_Brandon_PA1/Source Code$ ./server
Select operation mode (Server = 1, Peer = 2): 1
This process will operate as the Server.

Select the port that will be used for listening: 1111

The server is now active.
```

Figure 3: Using the Server

- 7) Navigate to each of the peer# directories (e.g. peer1) and open each of the “peer# program files.

This will require multiple open terminals. For each of the peers, select the IP address to be 127.0.0.1 (local host) and use the port the server is listening on.

A terminal window with a dark purple background. The title bar shows a window icon, a close button, and the text 'brandon@brandon-UX310UA:~/Desktop/01\_Dotson\_Brandon\_PA1/Source Code'. The terminal content shows a user prompt 'brandon@brandon-UX310UA:~/Desktop/01\_Dotson\_Brandon\_PA1/Source Code\$' followed by the command './peer1/peer1'. The program output includes: 'Select operation mode (Server = 1, Peer = 2): 2', 'This process will operate as a Peer.', 'Select the IP address to connect to: 127.0.0.1', 'Select the port: 1111', and 'Please enter a command (Use ? for help):' followed by a cursor. The terminal window has a small orange bar at the bottom right.

```
brandon@brandon-UX310UA:~/Desktop/01_Dotson_Brandon_PA1/Source Code$ ./peer1/peer1
Select operation mode (Server = 1, Peer = 2): 2
This process will operate as a Peer.

Select the IP address to connect to: 127.0.0.1
Select the port: 1111
Please enter a command (Use ? for help):
```

**Figure 4: Using the Peer**

- 8) If each of the peers prompt you to enter a command, you are ready to start.
- 9) Use the “quit” command to stop each of the peers when you are finished and the “close” command with the last peer.

## Command Listing

The following are the user commands for the peers:

**Registry:** The registry command registers all of a peer's files to the indexing server's file index. When doing so, the file name and the peer's ID number are stored (To use this command, type either "registry" or "r" then press enter).

**Search:** The search command searches the indexing server for a particular file. All of the owners of the file are listed by peer ID number (To use this command, type either "search" or "s" then press enter).

**Obtain:** The obtain command first uses the search command to verify a file is present. After this, the peer is provided the peer ID number of the owner of the file. This ID number is used to find the port the peer needs to obtain the file. The peer then connects to the owner of the file and copies the file over.

**Test:** This command is used for testing only. The peer begins a search test where a list of 1000 queries are checked against the file index. The elapsed time is output on the screen when this test is complete.

**Quit:** The quit command ends the session between the peer and the server. **Does not work properly.** (Type "quit" or "q" to use this command).

**Close:** This command causes the indexing server to close. **Does not work properly.** (Use "close", "c" or "close\_server" to use this command).