Report Project2 Fall 2017.

Names: B Radha Krishna GTID:903237039, Parth G Desai GTID:903222132

**Objective:**

This project implements a shared memory based file compression service and a client library. This service allows blocking (synchronous) calls, non-blocking (asynchronous) calls.

The client is an example process. While the library contains the functions which can be called for synchronous and asynchronous calls to the server.

**Segments:**

-Synchronous calls

The client will call the library "synccompress" functions and send it the two paths it obtained from user (the source file and the compressed file path)

The library function creates three shared memory segments and the function will send the two file paths to the server using message queues.

The first shared memory segment is for source file while the 2nd shared memory segment is for the compressed file. both these can be upto 1 Mb. The 3rd shared memory segment is of size 1 int and for the library to know that the compression is done.

The client will set the flag to 0.

Then the function will be waiting till the compression is done using the flag (the 3rd shared memory segment).

After the compression is done the function copies the contents of the 2nd shared memory segment to the file path provided.

Then the function will delete the shared memory segments and return to the client, when the client can resume working on other work.

-Asynchronous calls

The client will call the library "assynccompress function" and send it the two file paths as arguments.

The library function creates three shared memory segments and the function will send the two file paths to the server using message queues.

The first shared memory segment is for source file while the 2nd shared memory segment is for the compressed file. both these can be up to 1 Mb. The 3rd shared memory segment is of size 1 int and for the client to know that the compression is done.

Then the function returns to the client with the flag's address so that the client can monitor the flag and continue its work.

When the flag does shift to 1 the client will call up cleanup function.

The cleanup function will connect to the 2nd shared memory segments and copy its contents to the designated file, then delete the shared memory segments and return back to the clients for the client to continue its work.

-Server

The server will open a message queue and spin upon head of the queue.

When it does receive a request for compression the first compression request it receives will be worked upon. The others request will be queued up in the order they arrive.

The sever using the filepaths and projids will attach to the shared memory segments and read the data from the 1st shared memory segment.

Using snappy compress 's c port the server will compress the data and store it in the 2nd shared memory segment.

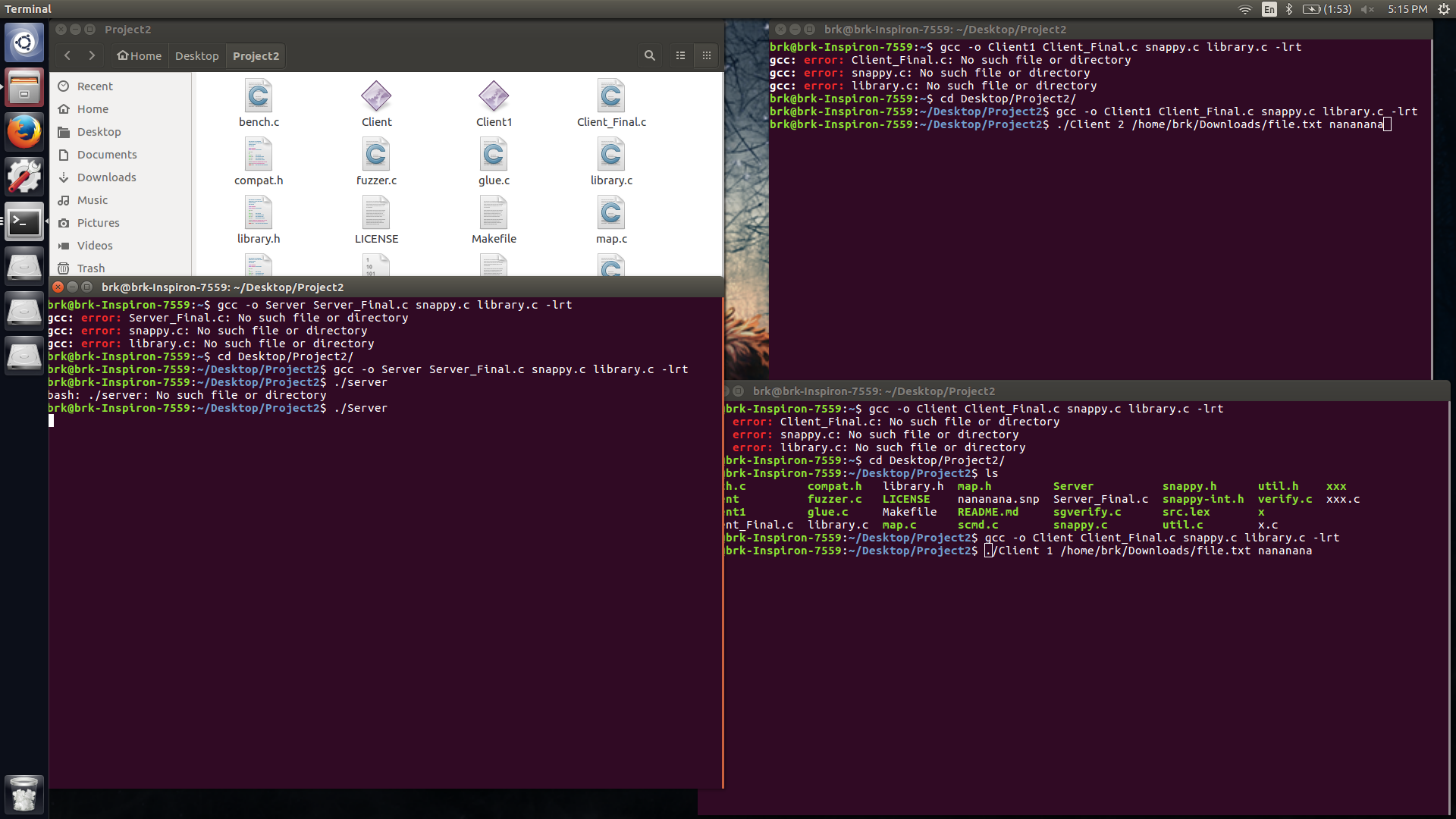
The server then updates the 3rd shared memory segment to 1. and detach itself from the memory segments and delete the particular queue entry and move onto the entry at the head or wait for one.

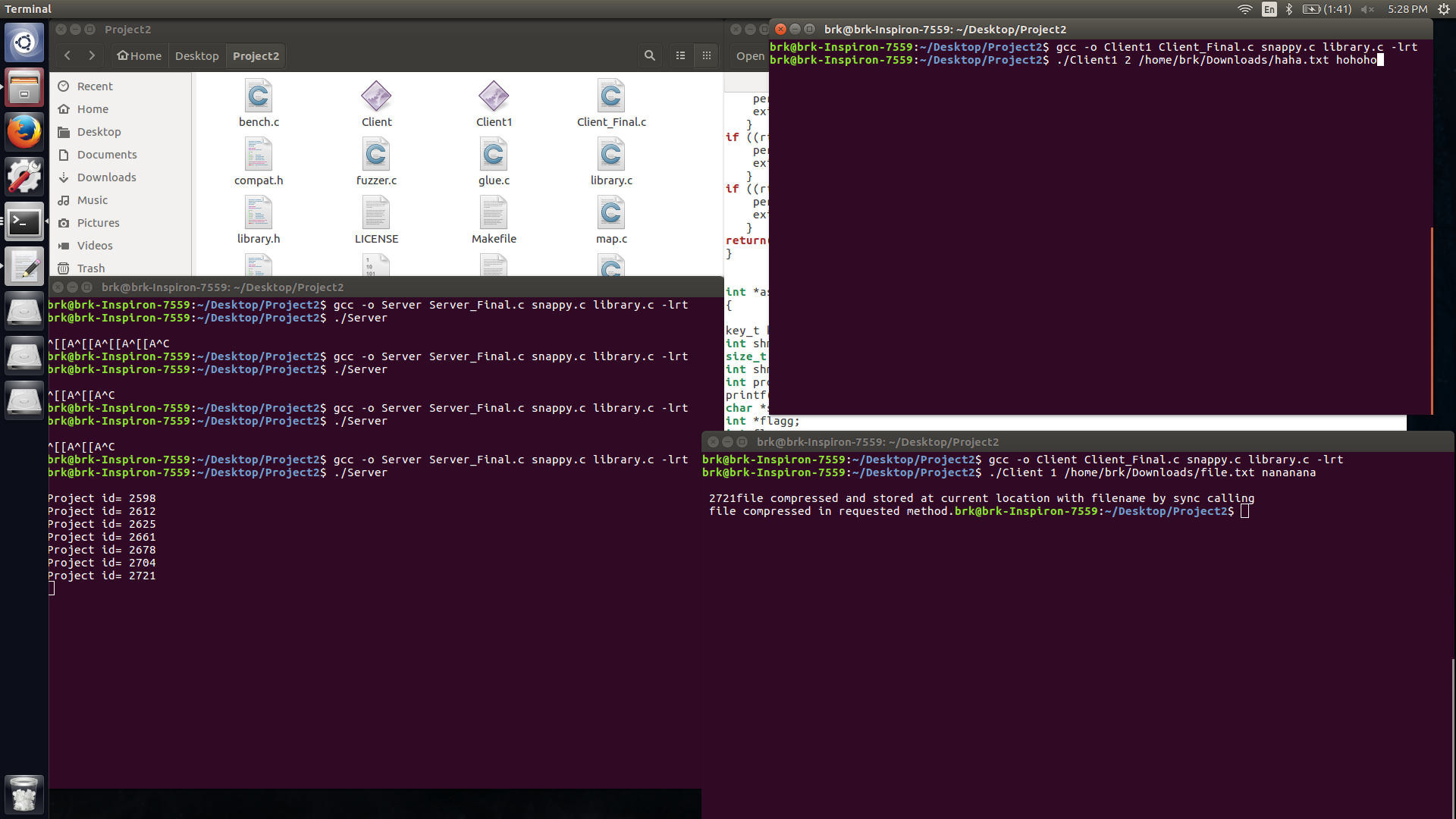
- The example client

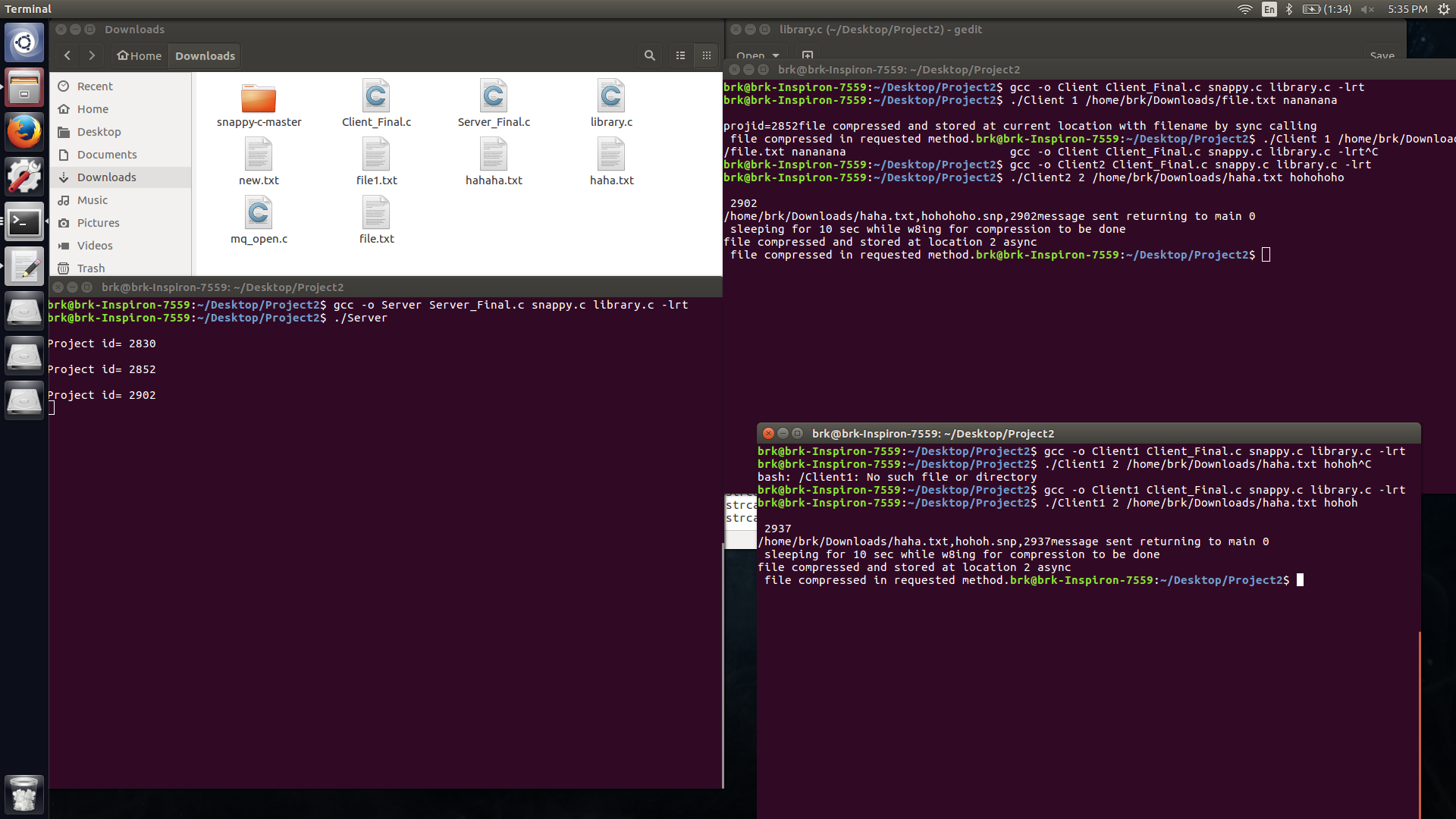
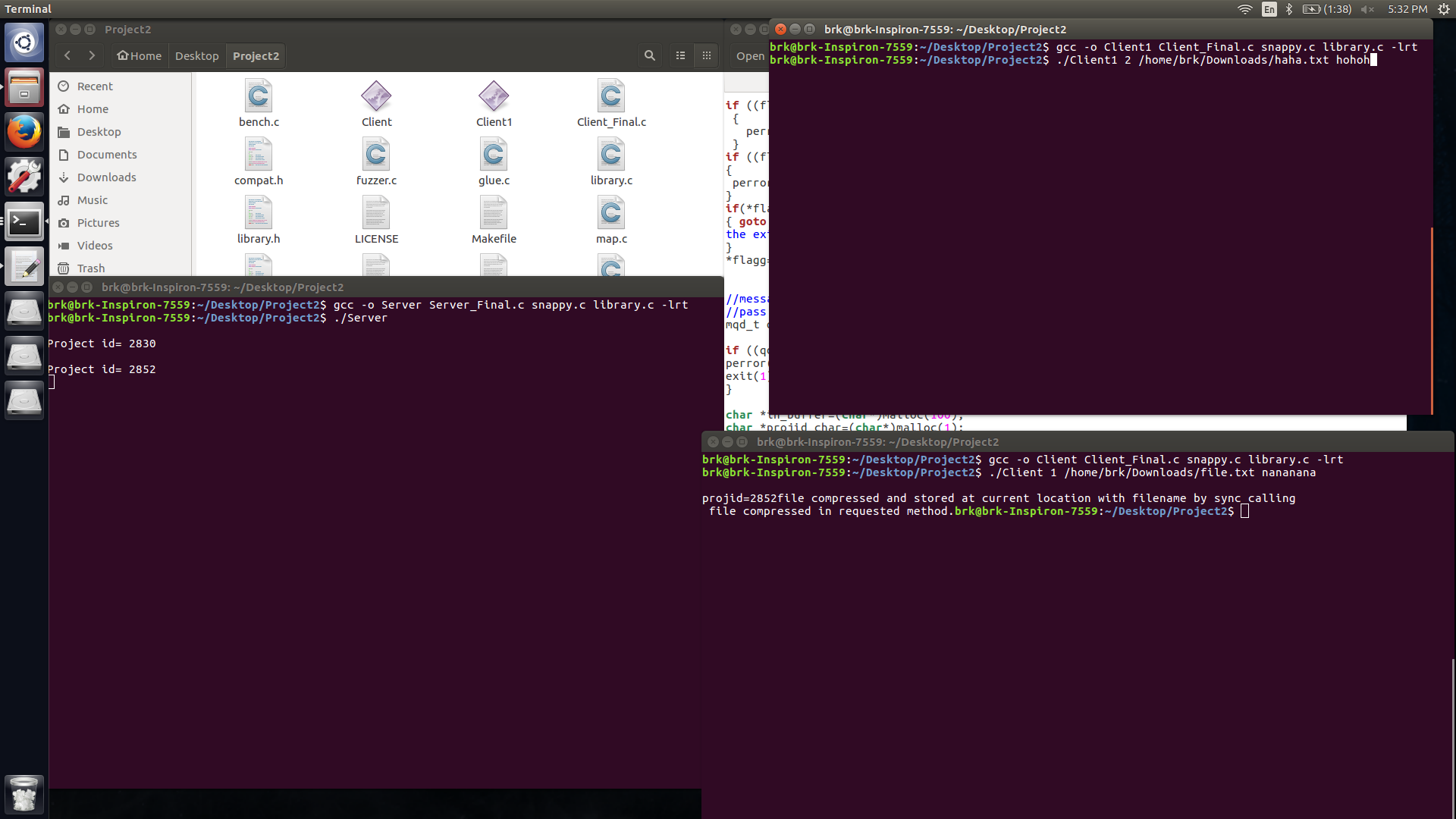
The client requires to count till greater than 1000 and compress the file given to it and check if the compression occurs correctly.

The client can call for both synchronous and asyncronous mode of compression.

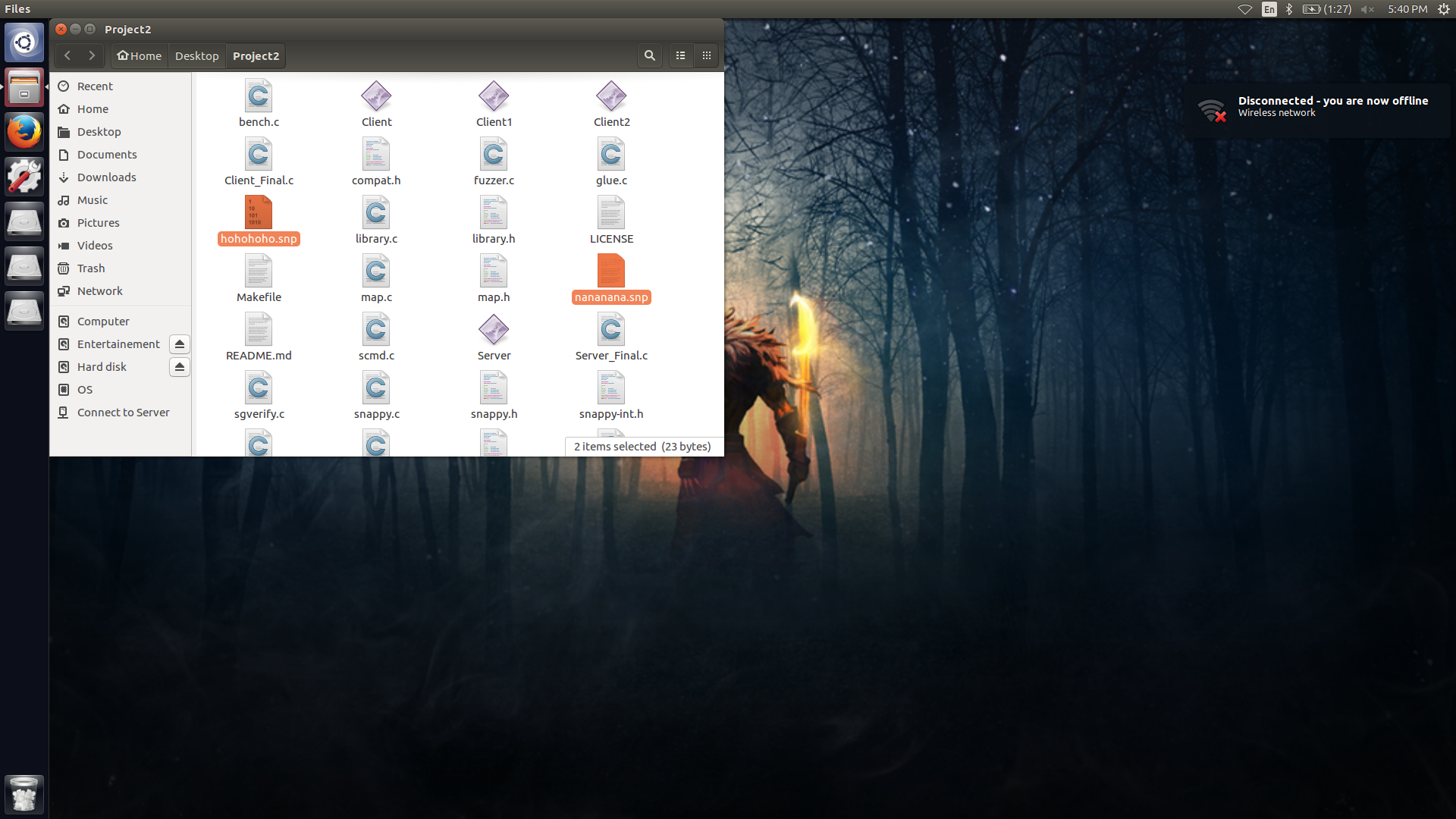
**Sample outputs shown**

**server started.**

**Server processing requested projid**

****

**Server being called in sync and async mode**

****

The compressed files created in the same directory