Lab Assignment 3 – Deadline 22nd March, 2014

For this assignment, there will be 3 entities as follows:

- Registry server: This will keep track of all instances of the FileServer

- File server: We will run multiple instances of the FileServer per node. For e.g., there could be 3 servers per node. Each FileServer will both a server and a client. It will register itself with the registry server.

- Client: The client will first contact the registry server to know ports of FileServers. For writes, it will send the FileWrite64K RPCs to the master. For reads, it can contact any FileServer.

This assignment will not deal with failures. So, the registry server will be started first. The FileServers will be started after that, followed by invoking the Client. Once the writes and reads are done, all servers and the registry server are killed.

<u>C version</u>

<u>Registry server provides the following APIs :</u>

bool RegisterServer(int port):
- *port* is the port on which the file server is listening
- Return value is as follows: true means caller has been made the master, and false means caller is the replica

*char * GetFileServers():*
- returns the ports of all the FileServers. The ports are returned as a "," separated string. The first port represents the master.

The behavior of the registry server will be like any server - it starts up, registers with portmapper, and waits for requests. (please check you have portmap installed on your system)

The FileServer behaves as follows:

- Takes the port as an input argument

- Modify main() of the server program as indicated in http://people.redhat.com/~rjones/secure_rpc/ Section 3.3 to make the server bind to the specified port.

- Use svctcp_create(), and svc_register()

- Calls RegisterServer(port) with the input port number to register this server with the registry server

- Use the return value to track whether this instance is master or a replica

- Call svc_run() to start the server

- Fail FileWrite64K() requests coming on a replica

<u>Client program does the following:</u>

- Invokes our registry to obtain all fileserver objects using the GetFileServers() API

- Parses the string to get the port number of the master and the replicas

- For each server, create the CLIENT objects using clnttcp_create(). The port number and IP of each server has to be provided as part of the first argument. Please remember we have multiple replicas running on the same host (no remote replicas).

- Client will end up with one CLIENT object per server, one for the master and one each for the replicas

- Writes can go to the master and reads to the replicas

<u>On a write request, the client does the following:</u>

- It gets the port number of the master as mentioned above

- Creates a CLIENT object for the server and issues FileWrite64k() RPCs as earlier in **assignment-2**

<u>On a read request, the client does the following:</u>

- It gets the port numbers of the master and each replica

- Sends an RPC to the master to get the number of chunks
    - *int GetTotalNumberOfChunks();*

- Creates n threads, where n is the number of FileServers created

- Each thread takes an integer i as a parameter

- Each thread creates a reference to the ith server

    - The thread then reads as follows:

ReadAlgorithm-1

```
// get a reference to the ith server
// In C, this is a call to clnttcp_create() using the ith port returned by the registry

while (true) {
 int chunkNum = getNextChunkNumber(); // this will atomically increment the chunk number and
return it

 if (chunkNum > totalNumberOfChunks) break;

 // issue a FileRead64K() for this chunk number
  FileRead64K(filename, chunkNum * 64K, data);

  // write the data read to the output file
  lock();
  fseek(ofp, chunkNum * 64K);    //chunk number starts from 0
  fwrite(ofp, data, 64K);              // write to filename  ~/Assignment3.out
  unlock();
}
```

Assignment Evaluation procedure:

1. Please Make sure you follow the following naming criteria

   - Registry server executable must be named as " registry "
   - Fileserver executable must be named as " fileserver "
   - Client executable must be named as " client "

2. Client program first writes the content of filename " Assignment3.txt " present in the home directory (~/).

3. Client then use FileWrite64k() RPCs to write the file data to Master.

4. Once client writes the file completely to the master , it then reads the file using multiple replicas as mentioned in ReadAlgorithm-1 (above). The final write should be in Filename " Assignment3.out " in home directory (~/).

Script we will be running so please follow above rules

1. ./registry
2. ./fileserver *port-1 registry-server-ip*
3. ./fileserver *port-2 registry-server-ip*
4. ./fileserver *port-3 registry-server-ip*

   (port-1, port-2, port-3 will not cause any bind exeption and these are the ports on which corresponding fileserver will listen)

5. ./client *registry-server-ip file-server-ip*

We will be crosschecking <u>your</u> client and fileservers with <u>our</u> fileservers and client. So please please follow all the above intsructions. Otherwise you may get zero for that.

Note: All fileservers will be on the same machine and registry and client may run on the different-2 machines.
References:
SUN RPC:

http://docs.oracle.com/cd/E19683-01/816-1435/rpcgenpguide-21470/index.html
http://linux.die.net/man/3/clnt_create
http://people.redhat.com/~rjones/secure_rpc/ - Look at Section 3.3

Upload Format:

If your **roll no**. is Student1234 , then create a diectory named as your roll number coantaining all your source codes and compress the folder into "Student1234_Cversion_Assignment3.tgz" and upload on the course portal.

The assignment evaluation will be automated. But, 50 randomly selected programs will be manually evaluated.
Also, we will be running MOSS. So, please do not copy.