Foreword

In this project, you will work with Remote Procedure Calls, specifically with Sun RPC. The RPC server will accept a jpeg image as input, downsample it to a lower resolution, and return the resulting image.

For extra credit, you may modify the code automatically generated by the rpcgen tool so that the RPC server is multithreaded.

Directions

1. Download the starter code in the tarball ud923-project4.tar.gz from the “Downloadables” section of one of project’s morsels the on the Udacity site.
2. Run ’tar -zxf ud923-project4.tar.gz’ to untar the file.
3. Begin programming by **modifying only those specified below.**
4. Turn in your code by copying and pasting onto the Udacity programming quizzes.

Building an RPC Server

Your first task will be to define the RPC interface through the XDR file **minifyjpeg.x**. The syntax and semantics of this type files are explained in the course videos as well as the documentation for rpcgen.

The rpcgen tool will generate much of the C code that you will need. It is recommended that you use the -N option for the “newer” rpc style that allows for multiple arguments. If you intend to make your server multithreaded, then you should also use the -M option.

In the generated file **minifyjpeg.h**, you will find the definitions of functions that you will need to implement. On the server side, you need to take the input image, reduce its resolution by a factor of 2, and return the result. A library that does this with ImageMagick is provided in the files **magickminify.[ch]**.

On the client side, the provided file **minifyjpeg_main.c** acts as a workload generator for the RPC server. It calls two functions which you should implement in the file **minify_via_rpc.c**.

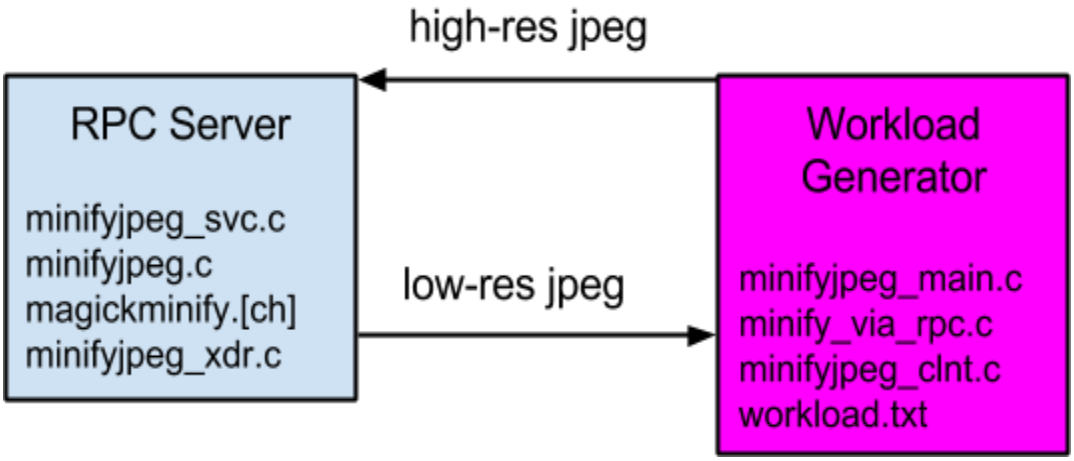
- *get_minify_client* - should connect to the rpc server and return the CLIENT* pointer.
- *minify_via_rpc* - should call the remote procedure on the client specified as a parameter.

In order to support images of arbitrary size and not worry about implementing packet ordering, the communication should use TCP as the transport protocol.

Here is a summary of the relevant files and their roles.

- **Makefile** - (do not modify) file used to compile the code. Run ‘make minifyjpeg_main’ and ‘make minifyjpeg_svc’ to compile the client and server programs.

- **magickminify.[ch]** - (do not modify) a library with a simple interface that downsamples jpeg files to half their original resolution.
- **minify_via_rpc.c** - (modify) implement the client-side functions here so that the client sends the input file to server for downsampling.
- **minifyjpeg.h** - (to be generated by rpcgen from minifyjpeg.x) you will need to implement the server-side functions listed here in minifyjpeg.c
- **minifyjpeg.c -** (modify) implement the needed server-side function here.
- **minifyjpeg.x** - (modify) define the RPC/XDR interface here.
- **minifyjpeg_clnt.c** - (to be generated by rpcgen from minifyjpeg.x) contains the client side code that executes the RPC call.
- **minifyjpeg_main.c** - (do not modify) a workload generator for the RPC server.
- **minifyjpeg_svc.c** - (to be generated by rpcgen from minifyjpeg.x) this contains the entry point for the server code. You may modify to make your code multithreaded. Otherwise, this is not recommended.
- **minifyjpeg_xdr.c** - (to be generated by rpcgen from minifyjpeg.x) contains code related to data structures defined in minifyjpeg.x
- **workload.txt** - (modify for your own testing purposes) an example workload file that can be passed into the client main program minifyjpeg_main.c



To turn-in this portion of the assignment, you should navigate to

https://www.udacity.com/course/viewer#!/c-ud923/l-3427108584/m-3924818685

and copy/paste your code in and submit. The site will run a "build verification test" that will help catch more obvious mistakes. A grader will review your work after the submission deadline.

Extra Credit

For extra credit, you may modify the server-side file **minifyjpeg_svc.c** so as to make the server multithreaded. Note that it is the *svc_getargs* function that copies global data from the rpc

library into memory that you control. This must be done *before* the function registered with svc_register returns. The function *svc_sendreply* can be called later.

You may also integrate the code with your Proxy Server so that it minifies all jpeg images before relaying them to the client.

References

Relevant Lecture Material

- P4L1 Remote Procedure Calls

RPC Material

- Sun RPC Example
- Sun RPC Documentation (maintained by Oracle; similar semantics, updated API.)
- From C to RPC Tutorial

Project Design Recommendations

- When testing and debugging, start with running small/light client workloads
- If your server is slow in responding, you may start timing out at the TCP socket and then the RPC runtime layer -- thus, look for options to change the timeout values used with RPCs

Rubric

RPC Client (50 points)

- RPC initiation and binding
- Correct communication with RPC server
- Multithreaded operation
- Proper clean up of memory and RPC resources

RPC Server (50 points)

- Interface specification (.x)
- Service implementation
- Accepts and processes RPC client requests
- Proper clean up of memory and RPC resources

Extra Credit (+10 points)

- Multithreaded RPC Server

Extra Credit (+5 points)

- Complete implementation of Getfile proxy server that downsamples jpeg images.

Questions

For all questions, please use the class forum so that TA's and other students can assist you.