

Remote Function Invocation

Joseph Cordina

February 23, 2008

This is the description for assignment 3 of unit CSA 2090, System Programming, for the year 2007/2008. This assignment is worth 20% of the total mark for this unit. The deadline for this project is 4th May 2008. You may do this assignment in a group of maximum 2 people, yet you may do this assignment individually also. A single mark will be allocated to the whole group, so choose your partner carefully and make sure that the work gets evenly distributed. Under no circumstances should code be shared outside your group. You will submit your assignment making use of the ASS system¹. Those of you without login access, please contact me on joseph.cordina at um.edu.mt. Each of you might be assigned a ten minute slot in which you will be presenting the implementation of your assignment and during which your program will be executed. I will run your program on two different systems (of different architectures) and if that works, you will pass your assignment. So really functionality is more paramount for this assignment, rather than design. The reason for this is to alleviate your load in terms of efficiency and due to the fact that you will have assignments for other credits that make use of socket programming. Please be reminded that you cannot copy and plagiarize to ease your way to a final submission. While you may discuss ideas with others, do not steal. You can find more information at

<http://www.cs.um.edu.mt/cs/links&resources/plagarism.html>

Read this information well since after wards no excuses will save you. Anyone resorting to such methods will be considered as trying to cheat and will even risk the removal from the degree program.

1 Submission Contents

You will submit your C files together with any accompanying header files (C language has to be used obviously) separately. You should also submit an example application but make sure you highlight those files which make part of the example. Please do not use any tar or zip file formats. Additionally include any libraries you might have used. The compilation can assume the presence of 'gcc' on which your system will be compiled. Include with your submission *makefiles* that can compile your system². You will need to provide two makefiles, one to generate the `client_stub_generator` and another for the `server_generator`. You can assume that the user will link his code manually with the generated stubs. I will try your code by building my own user code and configuration files and running separate instances on Free-BSD, SunOs and Linux machines in any order. With your submission, you will provide a file called report.txt which will be a very **brief** description of how your system works and provides the required functionality. Make sure you mention any special techniques or tools you made use of and also any considerations you have made to improve efficiency. More importantly, give details of the network protocol between the interacting processes. Please do not include code inside this file, since this is obviously not readable. Additionally, please note down any bugs

¹www.cs.um.edu.mt/~jcord/ass

²You can find plenty of tutorials on the net that teach you how to set one up

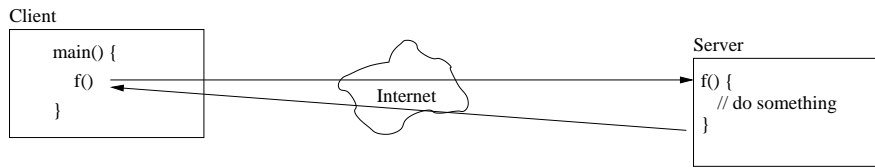


Figure 1: User's View of Function Call

your system might have. Listing bugs is common practice, and will benefit the marking process. This file should also contain references to resources you have made use of in your program. Finally, you will submit a file 'protocol.jpg' containing an image that describes in the most concise and clear manner possible your system's architecture and messages that are passed between the components.

Before submitting, please do clean up your code. Remove any dead code and useless comments. Every system calls output should be validated for errors and when appropriate, an error message reported to standard output. Also do place some comments to describe your structures in the code, at least to show that you understand how your own program works. Remember C is only unreadable if you make it unreadable. I cannot emphasize enough to be careful with your naming conventions and visual formatting.

2 Few Words and a Couple of Diagrams

As you know, invoking a function is a synchronous process. That is, you make the call, the caller waits until the callee returns from the function. This makes coding easier to reason about. But what if the function has to run on another machine on the other side of the world. Well the programmer should not care. So that is your assignment: to provide a way to allow a user to write a function (referred to as a *client function*) and will make a call to another function (referred to as the *remote function*) but this function is spatially away from the user. Thus for this assignment you will design a protocol (you have to use stream sockets aka TCP/IP sockets) to connect two programs together. The biggest problem is that we want the program to be as generic as possible (otherwise the solution is trivial no !!). So our aims are:

- The user should not have to change his code apart from splitting the remote function from the client function in separate files.
- The remote function can be anywhere in the world (as long as on the Internet).

So from the point of view of the user of your library the program should be reasoning as Figure 1.

To simplify your life (since then your assignment would be as big as an APT), we will only implement call-by-value. This means that if a function passes a pointer to anything, we will not carry the contents to the other side since we can never know the size of what we need to carry (languages like Java RMI solve this through serializable classes and reflection). Thus we can only pass things by value (structs, chars, ints, even pointer values) but not the value that the reference points to.

2.1 Configuration

Very simply, the system will work by the user filling in a configuration file specifying those functions that are remote. Then a number of operations will be performed on this configuration file. The client side will run an application that will generate the stubs for the client functions from the configuration. The remote side will run an application that will wait for requests for the functions in the configuration file.

The configuration file is in XML. Please do not parse the XML by hand but find open source libraries that can do this for you in C. There are plenty of good ones out there. An example configuration file follows:

```
<RFIDefinition> <!-- beginning -->
  <!-- List of headers that might be needed, mainly for type resolution -->
  <headers>
    <header>header1.h</header>
    <header>header2.h</header>
  </headers>

  <!-- List of Functions to be RMI'ed -->
  <rfifunction>
    <name>foo</name>
    <args>
      <type>int</type>
      <type>int</type>
    </args>
    <returns>long</returns>
  </rfifunction>

  <rfifunction>
    <name>foo1</name>
    <args>
      <type>long</type>
    </args>
    <returns>void</returns>
  </rfifunction>

  <!-- Server Details -->
  <ServerIP>127.0.0.1</ServerIP>
  <ServerPort>6345</ServerPort>

  <!-- Library name that implements the server size files -->
  <ServerImplLibrary>remote_lib1.so</ServerImplLibrary>
  <ServerImplLibrary>remote_lib2.so</ServerImplLibrary>

</RFIDefinition> <!-- end -->
```

So the configuration file above is stating that the user wants to remote function `foo` and function `foo1`. Function `foo` takes two arguments of type `int`, and returns `long`. Function `foo1` takes one argument of type `long`, and returns nothing. Also the definition up there, says that one should include `header1.h` and `header2.h` since they will be necessary for code generation (this is just an example and does not make sense). In real use, say a function takes a custom type, the definition of the type will be in a header file, so anyone working with the configuration file will have complete information on the types. We define the server ip address and the server port, on which the remote side will be running. And finally we define the name of the library that will contain the remote side implementation of the remote function³.

2.2 Client Function

On the client side, the build process will be as shown in Figure 2.

³As a hint, the first thing you should do is write a program that parses this config file and output from it types of pointers to these functions :)

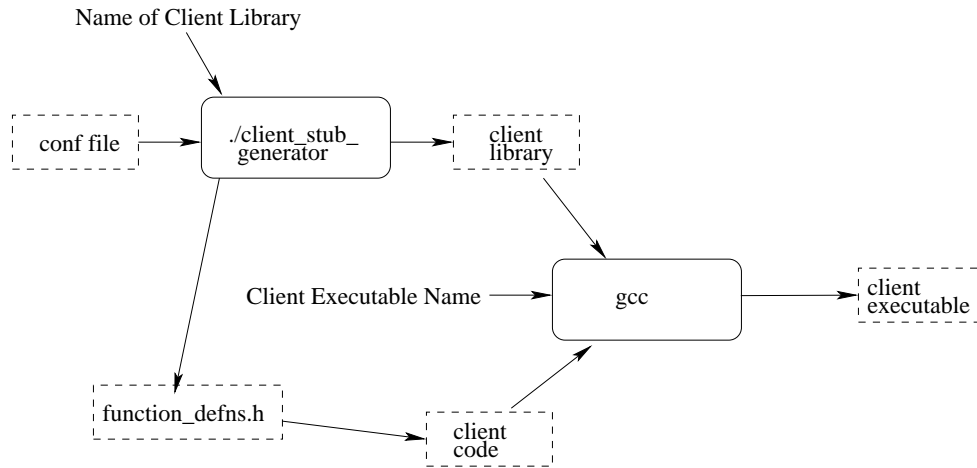


Figure 2: The build process on the client function side

So you will provide a program called `client_stub_generator` that reads a configuration file (name to be given as a parameter). This will generate code (!!) and will then be compiled to a library. To compile this to a library use `gcc -c` option and the `ar` utility for example

```
gcc -c client_stub.c -o client_stub.o
ar rcs libclient_stub.a client_stub.o
```

that will generate a library called `libclient_stub.a`. Thus when the user wants to link his code with the generated one, all he has to do is call

```
gcc -L. -l client_stub mainclient.c
```

where in `mainclient.c` there are the functions that make calls to the remote functions.

Now the stubs you have created for the remote function would open the socket to the server and communicate appropriately. Note that the user did not have to change anything in his `mainclient.c` file, and should work without modification (assuming all calls are call-by-value functions). Obviously running the final executable should report all the expected errors if there are problems with client-server connections, etc.

From the diagram, you can see that your application should also generate a header file. This will be called `function_defns.h` and is a commodity header file for the user to use in his code (say `mainclient.c` above). This can be used as a check that the tool expects the function definitions that the user had in mind.

2.3 Remote Function

Now the remote side will function as shown in Figure 3.

You will create an application called `server_generator` that will read the `conf` file and will generate a file called `server`. Upon running `server`, the application will open the appropriate ports and will listen on incoming requests. Now upon receiving the request it will make a call to user supplied remote function implementation. The only thing is that the function implementation does not need to be available when the server is run, as long as its available when the request is received. This gives us two things:

- If a user needs to change a remote function definition or add new remote function definitions, then he or she has to run the `server_generator` utility again and restart the server
- Yet if the remote function implementation changes, the server will not need to be restarted.

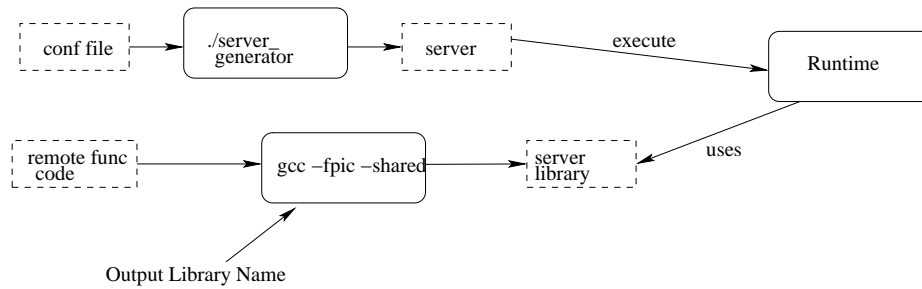


Figure 3: The build process on the remote function side

The way to allow this is to implement the remote side function in a dynamic library. It really sounds more complicated that it really is. Basically, if the user supplies you with a file in the right format, when the server receives a request, it will open this file using *dlopen* and get a pointer to the function using *dlsym*. Careful usage of these functions, will allow merely changing the library file to change the implementation without restarting the server.

The user will need to prepare this library though. This is done by compiling the code using `gcc -fpic -shared`. For example

```
gcc -fpic -shared remote_func_impl.c -o remote_lib.so
```

will take the remote function implementation and compile it into a library. The library name will be stored inside the configuration file (assuming directory where server is run) and thus easily found. Note that the configuration file does not need to be parsed by the server but only by the `server_generator` (this is to make the assignment easier).

2.4 So after all that

So with such a procedure it should be possible to have a generic implementation of a library allowing remote function invocation with no code changes necessary from the user's point of view. Note that this can be generalised further, but it is beyond the scope of this assignment. Also do not forget that we are dealing with call-by-value calls only, so your life is much easier with that assumption. Do not be scared by the fact that we are code generating a bit, after all code is just syntax until the reader gives it semantics.

3 What might matter

Most of the marks will be allocated to a functional system that allows multiple clients to make requests to multiple servers (note that a function can only be remote on one server). Thus I will build my own application, link it to your libraries, run the server and cross my fingers. If I get something back, it's an almost full marks !! It goes without saying that if you find a library implementing something similar (say RPC) you cannot use them. Obviously having a system that is free of race conditions, deadlocks and still considerably efficient would be nice. For your own good, first design your system before starting the coding. It will result in a better system and a quicker turn around time. Instances of the program will communicate with other instances through an Internet connection under non-privileged users.

This system has to be implemented using BSD stream sockets running on the TCP/IP protocol. Both the server and the client processes must be fault tolerant. They can both be terminated with a CTRL-C or the kill command. Ensure that proper cleanup is performed in each case. Thus the server must terminate all its child processes (if any) and close all open sockets. The child must close all its connections properly before

terminating. A client must be fault tolerant to broken connections. Ensure that you develop an appropriate protocol between the client and the server.

You can also assume that no deadline extensions will be given (as per usual really).

4 And To Conclude

Good luck and if you need anything, contact me on joseph dot cordina at um.edu.mt. Even better, make use of the discussion group to discuss ideas and ask questions. You can find a link to the discussion group on my website. Other students might help you out in the solution. Just make sure you do not post any code or given concrete solutions. And do not leave the assignment to the last week since you will definitely not make it in time!! Do it slowly and the ideas will mature.