Purpose: To gain an in-depth understanding of remote procedure calls.

You will build a remote program and a client that makes remote procedure calls to that program. Your remote procedure calls will use TCP.

Your remote program (server) will provide several remote procedures. Your rpc client will allow the user to call any of these procedures. For the remote program you must use your personal RPC program number. Compile and run the `Uniq_Id.c` program (from `~volper/classes/472/programs/rpc_example`) to get your program number.

Overview: You are writting an rpc (client/server) program for an automobile repair shop. When a customer brings a car for repair, information about that car is entered into a remote data base. This data base represents the queue of cars awaiting repair. When a technician becomes available, a car from the queue is given to that technician for repair (and that car is removed from the queue).

THE SINGLE MACHINE PROGRAM

In accordance with the RPC development procedure covered in class a single process (non-network, non-rpc) version of the program must be built first. Use the `repair_client.c` and `repair_services.c` programs found in the `programs/rpc` directory You should copy that program, compile and run it so you know what it does. The program is already separated into two files for your convenience. When building the RPC version of the program you must build the inteface procedures with the same interface for the services as in `repair_services.c`

You must make each of the services provided by the program available as remote procedures in a remote program.

The emphasis here is on the rpc implementation. Keep the same (simplified) user interface as I have provided.

THE RPC VERSION

Your first job is to define the interface or `.x` file. Make sure the number of values passed to each of the remote procedures corresponds to the number of parameters the corresponding sevice procedures requires. Make sure the number of values returned by each of the remote procedures corresponds to the number of values the corresponding sevice procedures delivers back to the user. Remember any service procedure parameter that is a pointer may be used to either send or return values. You have to analyze the code to determine this.

THE RPC SERVER PROGRAM

Since I give you the service procedures; your job is to implement the the server-side interfaces to provide the correct packing and unpacking necessary between the server procedures and the rpc calls.

THE RPC CLIENT PROGRAM

The rpc client main program will take one optional command arguement, the name of the host to which it is to make its rpc calls. This means copying in the Comer switch with the service option deleted into the client program. You also must add the call to create the handle to the server.

Your other job is to implement the the client-side interfaces to provide the correct packing and unpacking necessary between the client and the rpc calls.

Submit: The fully commented copy (print-out) of the source code for the following files: `repair.x`, `repair_cif.c`, `repair_sif.c`, `repair_client.c`, In addition the source code for these files should be placed in your home directory. Note: `repair_client.c` is going to be mostly my code, but I want to make sure you have made the modifications. `repair_services.c` should not be changed and should not be submitted.

Note that the files: `repair_clnt.c`, `repair_svc.c`, `repair_xdr.c`, `repair.h`, will be automatically generated, but since you don't do anything to them you should not submit them.

Reminder 1: Your parameters in `repair_client.c` *must* use the types and modes specified for the single machine program. Do not change the prototypes.

Reminder 2: Remember that `repair_sif.c` will need prototypes for each of the services functions. They should be the same as those in the `repair_client.c`.