CS 162 RPC Lab

RPC implementation

TABLE OF CONTENTS

Now that you have specified the protocol for the `ECHO` RPC, you will be implementing its actual functionality.

Example

To give you an idea of how RPC implementations look, an implementation for the `EXAMPLE` RPC has been provided to you in `server.c`.

```
int* example_1_svc(int *argp, struct svc_req *rqstp) {
  static int result;
  result = *argp + 1;
  return &result;
}
```

This code is what runs on the server after it receives an `EXAMPLE` RPC request from the client. As a brief rundown of what the code is doing, the function takes in a pointer to the value received from the client, dereferences it and adds 1 to the received value, then returns a pointer to this new incremented value. The return value is then serialized and sent back to the client by the stubs generated by `rpcgen`.

If you run `make`, you will see an error since you defined the `ECHO` RPC in the `KVSTORE` program specification but have not yet implemented the corresponding server-side function.

Implementing ECHO

Now implement the `ECHO` RPC in `server.c` so that it echoes the received message string back to the client. **After adding the `ECHO` RPC to `kv_store.x` and running `make`**, you should be able to find the function signature for `echo_1_svc`, the function you will need to add to `server.c` and implement, in `kv_store.h`. You may want to copy the existing `example_1_svc` function and change the arguments and return types appropriately to make sure you are following the correct convention.

In your implementation, you will first need to recover the message string from your RPC request in a similar manner as the `EXAMPLE` RPC, then pass the message string back into the reply. You shouldn't need to use `malloc` or `free` here, but you will need to place the pointer to the string in a static variable as shown in the `EXAMPLE` RPC stub (variables on the stack would be freed as soon as the function returns).

Before moving on, make sure that your code is now compiling with `make`.
