I use `clang-format -style=google` for style and `git` for version control.

How messages are transferred between server and client?

I adopted the HTTP Protocol standard that each message is prefixed with a header line indicating how many bytes the receiver should expect to get. Notice, due to the principle of separation concern, this message is purely a byte stream, we don't distinguish between whether it is a `struct rpc_request` or `stuct rpc_response`. (These two structs will be explained below)

A message will look like:

```
Message-Length:12

Hello World!
```

where the header line is separate from the real message content by `'\r\n\r\n'` (double carriage return + new liner)

How is rpc_request and rpc_response defined and serialized?

I paste my struct definitions here for clarity:

```
/* the struct for rpc request */
typedef struct {
  int command_op;      /* the procedure to call */
  int param_num;       /* how many parameters are packed */
  size_t* param_sizes; /* size of each param in bytes */
  char** params;       /* pointer to each param in bytes form */
} rpc_request;
```

Each `rpc_request` contains the procedure op to call and how many parameters are passed, and `rpc_response` contains the errno and how many return values are packed in this struct. Notice in both struct, they are able to pass variable number of parameters. A tacit understanding is assumed that which parameter corresponds to which argument. This is OK since we are the author implementing both the server side and client stub.

How does recursive treedirnode serialized?

The recursive nature of the `struct treedirnode` makes it a bit special in term of serialization. I adopt a Depth-first-traversal approach.

In a line-by-line-centric method, the first line will be the name of this node path, with the second line being how many subdirectories it has. Starting from the third line, it recursive to the first subdirectory of current node with all its children (if any). After the whole path down the first subdirectory is fully serialized, the program starts the second subdirectory serialization.

Let's see an example, suppose we have such a made `struct treedirenode`:

```
              hello
          /    |       \
        world  rpc     computer
       /
     network
```

Follow my DFS algorithm, after serialize this whole tree into a stream buffer, it looks like as follows:

```
(as usual, each line is end by a '\r\n' as in HTTP Protocol)
NodeName:hello
ChildNum:3
NodeName:world
ChildNum:1
NodeName:network
ChildNum:0
NodeName:rpc
ChildNum:0
NodeName:computer
ChildNum:0
```

How does server handle concurrent clients?

Our server side use multi-process model to handle concurrent clients with `fork()` sys calls. The main server process keeps accepting new client, and for each new client `fork()` out a new service process to take care of the client from beginning to end. And the main server process periodically harvest zombie child processes to release operating system resources.