Ray Imber
CS470
Assignment 2

Client / Server Architecture

Build and Run Instructions:
There is an included makefile.
To build:

untar the file
then type:

```
make clean
make
```

the following files will be created:
./server.o
./client.o
./services/f.service
./services/g.service

To Run:
Run the first server:
The server takes as it's first parameter a port, or a hostname and port to use to identify itself:

Example:
./server.o www.domain.com:3030
This will start server.o on port 3030 with www.domain.com as it's personal host name.

or:
./server.o 3030
In this configuration the server will start on port 3030, and it will attempt to determine it's personal host name from the operating system.

Run the second server:
After the first parameter, and other parameter will be treated as a hostname of another known server:

Example:
./server.o www.something.com:3030 www.alt-server-1.com:2025 www.alt-server-2.com:4039
This will start the server on port 3030 with www.something.com as it's personal hostname.
The server will then use www.alt-server-1.com on port 2025 and www.alt-server-2.com on port 4039 as alternate servers.

The Client:
The client takes a host string as it's single parameter, to use as a primary server:

Example:
./client.o www.something.com:3030

The client will use www.something.com on port 3030 as it's primary server.

The Protocol:
I chose to implement a custom protocol of my own design instead of the example protocol.
I wanted to design a protocol that was slightly more compact and modular, because I wanted to create a parsing library that could be shared between the client and server. I also wanted a very small set of delimiting tokens to make the parsing as simple as possible.
My set of delimiters:

- the new line character, "\n",
- the comma, ","
- the ampersand, "&"

There is a header portion that describes the message, and then any number of inner messages that can be the result of sub programs or services. This made the creation of a plugin based services easier to implement, because they could just directly append their results to the server output without interpretation or interference from the server.

Implementation:
Server:
The server was implemented with a custom hash table and linked list library. Local services are based on a plugin architecture using popen. The server will run up any program in the ./services directory that has a .service extension. The server does not interpret the output of the service in any way. The service promises to output a correct protocol string that will be appended to the server output.

Client:
The client uses a linked list of servers to query. The client attempts to try all servers in it's list until it gets a response or runs out of servers to try. This list starts with just the primary server, and then dynamically grows based on the receipt of alternate servers. After each request, the list is reset to just the primary server.

Appendix: Protocol Examples:

Server Hello message:

```
SERVER/HELLO
SERVER/INFO&www.something.com:4000
SERVICE/INFO&f,3,arr
SERVICE/INFO&g,2,arr
```

Server Unknown Service:

```
SERVER/UNKNOWN
SERVICE/INFO&f,2,arr
SERVER/INFO&www.secondary1.com:4010
SERVER/INFO&www.secondary2.com:5000
```

or

```
SERVER/UNKNOWN
SERVICE/INFO&f,2,arr
SERVER/INFO&none
```

Server Mismatched arguments:

```
SERVER/RESULT
SERVICE/MISMATCH&2
SERVICE/INFO&f,3,arr
```

Server Busy:

```
SERVER/BUSY
SERVICE/INFO&f,2,arr
SERVER/INFO&www.secondary1.com:4010
SERVER/INFO&www.secondary2.com:5000
```

Server Error:

```
SERVER/ERROR&(Optional error msg to present to the client)
```

Server Result:

(an array of ints.
 type = arr
 len = number of elements in the array.
 Elements are delimited by commas.)

```
SERVER/RESULT
```

SERVICE/RESULT&arr&10&1,2,3,4,5,6,7,8,9,0

or

(raw binary data.
 type = raw.
 len = number of bytes that make up the result.
 It is assumed the client knows how to interperate the data.)
SERVER/RESULT
SERVICE/RESULT&raw&256
(data)......

Service announce message:
SERVICE/INFO&f,3,arr

Service Error message:
SERVICE/ERROR&(Optional error message);

Client Request message:
CLIENT/REQUEST&f,4,5,6