CIS 620 Project 3 Fall 2018

(Due: Nov. 19)

The purpose of this project is to familiarize yourself with the TCP/UDP socket (in C and Go) and the advanced UNIX file operations such as `lseek`, `lockf`, etc. You are asked to write a client-server program in which the client can contact the db server to access a database.

You also need to write a service-map program which resides at a well-known UDP socket. Each group should should use a five-digit number which is a "2" followed by the last four digits of your student id. The service-map server (written in Go) accepts registration from the db server (written in C) and advertises socket addresses to the prospective client (written in Go). When a database server starts, it first requests a local TCP port. Then, it broadcasts a registration message in UDP. For example,

```
PUT CISBANK tcp-portnum
```

The service-mapper will receive this message, store the service name CISBANK along with the IP address and port number of the db server in its table, and reply an ASCII string "OK" to the db server. When a client starts, it broadcasts a request message:

```
GET CISBANK
```

The service-mapper will return a datagram containing the information of the database server (i.e. dotted IP and port number in ASCII). An example of the datagram is shown below

```
137.148.204.16:8765
```

You need to copy the database db18 from the directory ~cis620s/pub to your working directory. Each record in the database has the following fixed format:

```
struct record {
    int   acctnum; /* unique key in sorted order */
    char  name[20];
    float value;
    int   age;
}
```

The field value is stored in binary (not ASCII) format. You can use the command od if you want to see the file content.

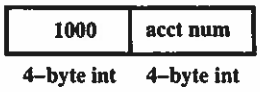
The following are the two commands which can be issued from a client to a database server:

```
query acctnum
update acctnum amount
```

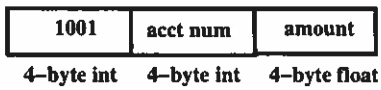
The query command requests the server to get and reply the record for a person. The update command asks the server to add an amount to a person's record in the database.

The client uses the following message formats to communicate with the server:

query message format



update message format



Both of the integer and float fields are transmitted in big-endian order. The server returns the result via an ASCII string and then closes the TCP connection.

When the server establishes a connection with a client, it forks a child process to handle the request. The child process has to use the operations `lseek` and `lockf` in UNIX. The `lseek` function can locate the position to be rewritten in a file, while the `lockf` can lock a file (or even a record) to prevent multiple clients updating the same record simultaneously. The parent process should use a signal-catching routine to clean up any child process which has terminated. The following is an example to run the service mapper on arthur, the database server on the machine bach and the client on chopin.

```
arthur% ./servicemap
Received from 137.148.204.16: PUT CISBANK 8765
Received from 137.148.204.10: GET CISBANK

bach% ./server
Registration OK from 137.148.204.15
Service Requested from 137.148.204.10
Service Requested from 137.148.204.10

chopin% ./client
Service provided by 137.148.204.16 at port 8765
> query 11111
MULAN HUA 11111 99.9
> update 34567 8.2
BILL SUN 34567 74.2
> quit
```

Turnin

Each group (at most two students) has to submit this project electronically using the following turnin command on grail:

```
turnin -c cis620s -p proj3 client.go server.c servicemap.go makefile
```

Each group also needs to hand in a hard-copy document which includes the description of the protocol, the code, experiences in debugging and testing, etc. The cover page should contain your picture(s), name(s) and the login id you used to turnin the project. Start on time and good luck. If you have any questions, send e-mail to `sang@eecs.csuohio.edu`.