Assignment #5 - Device Driver Network Support
CMPSC311 - Introduction to Systems Programming
Fall 2013 - Prof. McDaniel

**Due date: December 13, 2013 (11:59pm)**
**NO EXTENSIONS OR LATE SUBMISSIONS ACCEPTED**

In this assignment you will extend your device driver to use a network connection to send and receive commands and data with the disk array. Please read the following instructions carefully and perform the requested steps as directed. Note that a superficial reading of this assignment may lead you to believe that the work is easy, but it is not. Please allow ample time to complete the assignment, as no extensions or late submissions will be accepted.

**SMSA Network Specification**

You are to implement the client side of a network protocol for the SMSA disk array. The idea of the new architecture is to have all requests be transferred over a network to a server and responses returned. In this way, the program could support using a disk array placed anywhere on the Internet from your host computer. Programmatically, all calls to `smsa_operation()` should be replaced by sending a message to the `smsasvr` process provided to you and receiving the response. The code to perform the send and receive is to be written by you in the `smsa_client_operation` function.

The protocol used to by the SMSA system consists of two messages. The *SMSA request message* is sent from your client program to the server and contains an opcode and buffer (where needed). The *SMSA response message* sent from the server back to your client containing a result code and a buffer where needed. Both messages use the same format:

**SMSA Packet definition**

| Position | Name | Description |
|---|---|---|
| Bytes 0-1 | length | how many total bytes in packet |
| Bytes 2-5 | opcode | the opcode for the command |
| Bytes 6-7 | return | return code of comamnd |
| Bytes 8-263 | block | as needed, SMSA_BLOCK_SIZE |

To implement the SMSA client, implement `smsa_client_operation( uint32_t op, unsigned char *block )` - This is the client operation. If not already connected, it should connect to the `SMSA_DEFAULT_IP` address and `SMSA_DEFAULT_PORT` port. Once connected it should send a request message described above to the server and wait for the response. The response will have the return code and an optional block of data. The opcodes are exactly the same as we have seen in previous assignments.

The same simulator program you used from last assignment will be used, as well as a new server program. The code provided reads a *workload* file of virtual read commands and calls your implementation to serve then. The client program is called as follows:

```
USAGE: smsaclt [-h] [-v] [-l <logfile>] [-c <sz>] <workload-file>

where:
    -h - help mode (display this message)
    -v - verbose output
    -l - write log messages to the filename <logfile>
    -c - set cache size to <sz> lines

    <workload-file> - file contain the workload to simulate
```

The program will print out a good deal of output if you use the `-v` verbose flag. Use this to debug your program. It is also encouraged to use the logging facility provided by the `logMessage` function listed in the

cmpsc311_log.h file. See its use in the simulator code for examples. To redirect the output into a utility you can use to walk through the data, redirect the output to less.

Note that you will also have to have a server running as well. The server program is run as follows:

```
USAGE: smsasrvr [-h] [-v] [-l <logfile>]

where:
    -h - help mode (display this message)
    -v - verbose output
    -l - write log messages to the filename <logfile>
```

**To do:**

1. From your virtual machine, download the starter source code provided for this assignment. To do this, download the starter code linked to the announcements section of the course webpage (front page). Decrypt, untar, and unzip the repository using the gpg and tar tools as you did for the previous assignment. (You will need to use the password in the related slides to decrypt the file).

2. You are to use your own code from the previous assignments (copy the driver and cache code into the assign5 directory). Although I will provide my code later for instruction only (so you can fix any problems you had with your code). I will not grade any assignment that uses my code for the driver.

3. You are to implement smsa_client_operation defined in the interface header file smsa_network.h whose implementation is in smsa_client.c defined above. Note that I am providing sample code for the server in smsa_server.c. YOU ARE NOT ALLOWED TO COPY OR USE THIS CODE, BUT CAN USE IT TO FIGURE OUT WHAT YOU NEED TO DO.

4. You are to integrate calls to the network functions as needed in your driver code.

5. Note that you may need to define a number of other support functions to make clean code here. All of these functions should check the correctness of every parameter and log an error and return -1 if any value is illegal.

6. Run the program with the example workload files and confirm that they run to completion. Sample output will be provided to compare against your run.

**To turn in**:

1. Create a GPG encrypted tarball file containing the assign5 directory, source code and build files as completed above. Email the program to mcdaniel@cse.psu.edu and the section TA (listed on course website) by the assignment deadline (11:59pm of the day of the assignment). The tarball should be named LASTNAME-PSUEMAILID-assign5.tgz.gpg, where LASTNAME is your last name in all capital letters and PSUEMAILID is your PSU email address without the "@psu.edu". For example, the professor was submitting a homework, he would call the file MCDANIEL-pdm12-assign5.tgz.gpg.

   Use the same password you used to decrypt the original file to encrypt this file.

2. Before sending the tarball, test it using the following commands (in a temporary directory – NOT the directory you used to develop the code):

   ```
   % gpg LASTNAME-PSUEMAILID-assign5.tgz.gpg
   % tar xvzf LASTNAME-PSUEMAILID-assign5.tgz
   % cd assign5
   % make
   ... (TEST THE PROGRAM)
   ```

3. Send the tar file in an email with the Subject Line **CMPSC311 ASSIGNMENT #5** to the professor and your section TA, with yourself CCed. YOU ARE RESPONSIBLE FOR DOING THIS CORRECTLY. Lost emails, emails not sent to both the professor and TA, bad tar files, bad encryption are your fault and it will be treated as late assignments. There will be no exceptions.

---

**Bonus Points**: Measure the RTT (round-trip time) of each message and keep a moving average of the time of each kind of operation (reads, writes, seeks) and log it at the end of the run using the `LOG_OUTPUT_LEVEL`.

---

**Note:** Like all assignments in this class you are prohibited from copying any content from the Internet or discussing, sharing ideas, code, configuration, text or anything else or getting help from anyone in or outside of the class. Consulting online sources is acceptable, but under no circumstances should *anything* be copied. Failure to abide by this requirement will result in dismissal from the class as described in our course syllabus.