Assignment #4 - LionCloud (v 1.2)

<div>提交作業</div>

截止日期 5月1日，23:59 　　　**分數** 125 　　　**正在提交** 文檔上傳 　　　**文檔類型** tgz以及tar.gz
可用 4月1日 0:00 - 5月1日 23:59 約 1 個月

In this assignment you will extend the device driver you have been building in the previous assignments. Everything about the LionCloud device remains as before, except as described in the specification here. At the highest level, you will extend the code to support networking communication to a server program that implements the LionCloud devices. All of the extensions will be made to the functions modified in the previous assignments as well as a new file `lcloud_client.c`.

Network Attached Lion Cloud

For this assignment, the operation of the LionCloud system is separated into two programs; the `lcloud_client` program which you will complete the code for, and the `lcloud_server` program which is provided to you. To run the program, you will run the server in one window and the client in another. You will use a local network connection (called the loopback interface 127.0.0.1) to connect the two programs running on the same machine (yours).

The challenge of this assignment is that you will be sending your lcloud requests and receiving responses over a network. You must start with your code from assignment #3, as it requires it. The assignment requires you to perform the following steps (after getting the starter code and copying over needed files from your implementation of the previous assignments).

The first thing you are to do is to replace the all of the calls to IO bus request in your code to:

`LCloudRegisterFrame client_lcloud_bus_request( LCloudRegisterFrame reg, void *buf );`

This function is defined in the new file `lcloud_network.h` and performs the same function as the original bus request function it is replacing.

The remainder of this part of the assignment is your implementation of the `client_lcloud_bus_request` function in the `LCLOUD_client.c` code file. The idea is that you will coordinate with the a server via a network protocol to transfer the commands and data from the client to the server. The server will execute the commmands and modify the device storage accordingly.

The first time you want to send a message to the server, you have to connect. Connect to address `LCLOUD_DEFAULT_IP` and port `LCLOUD_DEFAULT_PORT`.

To transfer the commands, you send the request values over the network and receive the response values. For each command the client will send the 64-bit register frame value (in network byte order) to the server. If the request has a non-NULL buffer (e.g., the operation is a `LC_BLOCK_XFER, LC_XFER_WRITE`), then the `LC_DEVICE_BLOCK_SIZE` buffer will be sent over the network immediately following the register value.

The server will respond to each command with the 64-bit frame register return value (in network byte order). If the request has a non-NULL return frame (e.g., ope code is `LC_BLOCK_XFER, LC_XFER_READ`), then the frame will be sent over the network immediately following the registers.

Note that you need to convert the 64 bit register values and Length values into network byte order before sending them and converting them to host byte order when receiving them. The functions htonll64 and ntohll64 are used to perform these functions respectively. The buffers are not numeric data, and thus don't need to be put in network byte order.

You will use the network APIs described in the associated networking lecture to complete this assignment. Note that the last thing you should do in your client program after a `LC_POWER_OFF` is to disconnect.

I have provided psuedo-code for the implementation of the network io bus in the file `assign4-psuedocode.txt` 📄 in files section of canvas.

Honors Option

For the honors option, you must implement encrypt the buffers before sending to the memory device and decrypt it when is returned. You should use the `gcrypt` library using the AES 128-bit cipher and a random key generated at run time.

Assignment Details

Below are the step by step details of how to implement, test, and submit your device drivers as part of the class assignment. As always, the instructor and TAs are available to answer questions and help you make progress. Note that this assignment is complicated and will likely take even the best students a substantial amount of time. Please plan to allocate your time accordingly.

1. From your virtual machine, download the starter source code provided for this assignment. To do this, go to the canvas website and download the file assign4-starter.tgz (you should be able to click the link **here**).

2. Create a directory for your assignments and copy the file into it. Change into that directory.

   `% mkdir cmpsc311`

   `% cp assign4-starter.tgz cmpsc311`

   `% cd cmpsc311`

   `% tar xvzf assign4-starter.tgz`

   `% cd assign4`

   Once unpacked, you will have the starter files in the `assign4` directory. All of your work should be done in this directory.

3. Copy over your file system and cache implementation file from the previous assignment. You are to complete the `lcloud_client` program by extending the functions described above in the `lcloud_filesys.c`, `lcloud_cache.c`, and `lcloud_client.c` source code files. You are free to create

any additional functions that you see a need for, so long as they are all in the same code file as they are used in.

4. Add comments to all of your files stating what the code is doing. Fill out the comment function header for each function you are defining in the code. A sample header you can copy for this purpose is provided for the main function in the code.

5. To initially test your program, you will run it with sample workload files provided (note that for the assignment the manifest and workload files are in the `workload` subdirectory of the assignment). The first step is to start the server program in a second terminal window. To do this, run the server program from the command line with sample input. E.g.,

`./lcloud_server -v workload/cmpsc311-assign4a-manifest.txt`

To run your client, run the program in a separate window from the command line with workload input. E.g.,

`./lcloud_client -v workload/cmpsc311-assign4a-workload.txt`

You must use the "-v" option before the workload filename when running the simulation to get meaningful output (but the program should work find without the -v option). You may wish to use the "-l [logile]" option to redirect your output to a file of either the client or server. Once you have your program running correctly, you see the following message at the end of the output:

LionCloud simulation completed successfully!!!

6. Once you have the code working for the first workload, you will need to run the programs with each of the remaining manifests and workloads. There are five pairs of workloads you need to run together

   1. `./lcloud_server -v workload/cmpsc311-assign4a-manifest.txt`
      `./lcloud_client -v workload/cmpsc311-assign4a-workload.txt`
   2. `./lcloud_server -v workload/cmpsc311-assign4b-manifest.txt`
      `./lcloud_client -v workload/cmpsc311-assign4b-workload.txt`
   3. `./lcloud_server -v workload/cmpsc311-assign4c-manifest.txt`
      `./lcloud_client -v workload/cmpsc311-assign4c-workload.txt`
   4. `./lcloud_server -v workload/cmpsc311-assign4d-manifest.txt`
      `./lcloud_client -v workload/cmpsc311-assign4d-workload.txt`
   5. `./lcloud_server -v workload/cmpsc311-assign4e-manifest.txt`
      `./lcloud_client -v workload/cmpsc311-assign4e-workload.txt`

   Note that there is a sixth manifest and workload you will not receive but be graded against as well. Expect that this last setup will test the limits of your code.

To turn in:

1. Create a tarball file containing the `assign4` directory, source code and build files as completed above. Submit the program before or on the due date through the canvas system. Because we have to grade this assignment before the end of the semester, THERE WILL BE NO EXTENTIONS OR LATE PENALTIES SUBMISSIONS ACCEPTED. As in the previous The

tarball should be named `LASTNAME-PSUEMAILID-assign4.tgz`, 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-assign4.tgz`. Any file that is incorrectly named, has the incorrect directory structure, or has misnamed files, will be assessed a 10% penalty.

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

```
% tar xvzf LASTNAME-PSUEMAILID-assign4.tgz
% cd assign4
% make
... (TEST THE PROGRAM)
```

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 dismissal from the class as described in our course syllabus.

7.