CS149 Project #1

By Josh Dittmer

Build Instructions:

1. Run the included build script
    a. Execute ./build.sh
    b. If permission is denied to run the script, execute chmod 777 build.sh first

OR

1. Build “mms.cpp” to make shared library “mms.so”
    a. Execute gcc -g -fPIC -c mms.cpp
    b. Execute gcc -g -o libmms.so -shared mms.o
2. Build “mmc.cpp” to make the memory console executable
    a. Execute gcc -L. -o mmc mmc.cpp -lmms -Wl,-rpath=./
    b. Run the memory console by executing ./mmc [memory size] [boundary size]
    c. For example, ./mmc 512 32
3. Build “test1.cpp” and “test2.cpp”
    a. Execute gcc -L. -o test1 test1.cpp -lmms -Wl,-rpath=./
    b. Execute gcc -L. -o test2 test2.cpp -lmms -Wl,-rpath=./

Note:

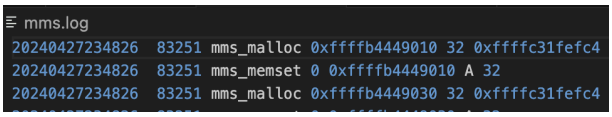
- The tests will pause halfway through. Enter a character to make them continue (done so you can check the state of the memory in the MMC).
- For the tests to run properly, make sure that the memory console is running first!

MMS Usage:

1. Use mms_init to attach to the shared memory region. Call this function before using any other mms functions
2. Use the mms_malloc function to allocate a new memory region. It will return a valid pointer exclusive to the calling program
3. Use mms_memset to set the data in a memory region. The pointer argument must be a valid pointer returned from mms_malloc or else the function will give an error. The pointer must also have been generated by the calling program, or the function will detect mismatched PIDs and give an error.
4. Use mms_memcpy to copy data from src_ptr to dest_ptr. This function can copy data in the following ways:
    a. From an mms memory address to an mms memory address
    b. From an external address to an mms memory address
    c. From an mms memory address to an external address
    d. It cannot copy from an external address to an external address (as per instructions given in class)
5. Use mms_print to print the data from a memory region. This function can also print data from external pointers.
6. Use mms_free to free a memory region.

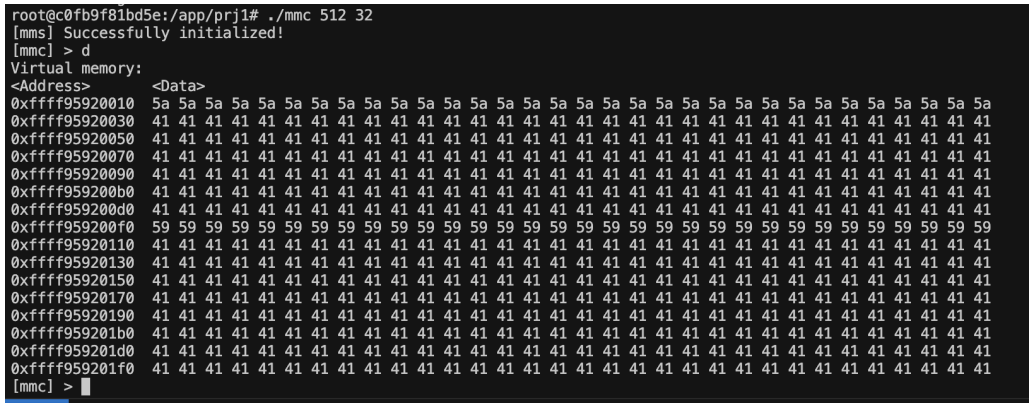
a. Note: mms_free will free a memory region and add it to the free list. However, mms_malloc does not check if adjacent memory regions are free, so mms_malloc will only ever be able to reallocate the original size of the freed memory region. This means that if boundary_size = 32 and memory is entirely filled with 32 byte regions and then freed, mms_malloc can only allocate a 32 byte region. Trying to allocate a 64 byte region, for example, will fail, even though there's actually enough space for it.

7. Use mms_cleanup before exiting to detach from the shared memory region
8. Running any mms function will log the function call in mms.log
    a. Example:

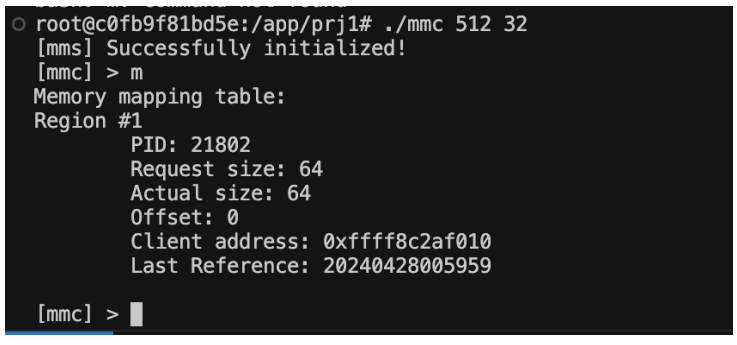


MMC Usage:

1. Run the memory console before executing any test programs
2. Commands:
    a. D [filename]
        i. Dumps the data in memory to the console in hex, and optionally to a file
        ii. Output (after running test 2):
        iii.



    b. M [filename]
        i. Displays the memory region table, and optionally writes it to a file
        ii. Output (halfway through test 1):

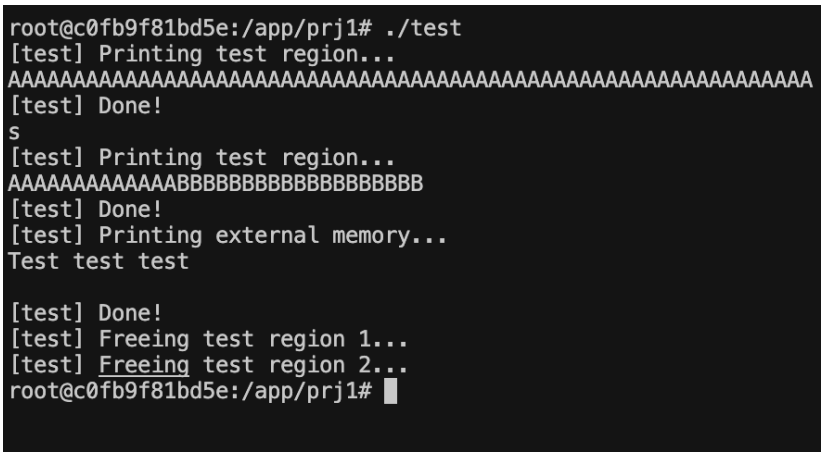


    c. E
        i. Exits the memory console

3. The memory console will create a file for the shared memory map (mms_file)

Expected Output of Tests:

1. Test #1:
    a. Output:



2. Test #2:
    a. Output: