**CSE 530**

**Assignment 1 Readme**

**Submitted by**

Gautam Sadarangani

1209570840

Prameet Singh Kohli

1209465527

**Part** **1:**

A header file “commonData.h” is included in both the main.c and ht530\_drv.c. It has the common struct definitions.

Steps to run:

The Makefile does the building of the module, the main program and also inserts(insmod) the module.

1. Run terminal and **cd into the project folder**.
2. To build the part **run** the command ***‘sudo make’*** in the terminal. Sudo is necessary as the makefile also inserts the modules.
3. Run the main file by **‘sudo ./main’**. Sudo is necessary since real time thread priorities require root privileges.

Additional Comments:

In order to correctly see the thread priorities, run **‘sudo ./main\_nosleep’.** This file does not have a random sleep between two consecutive file operations and user can observe that thread1 starts and finishes entirely before thread 2 starts and so on. The priority of threads are as follows:

Thread1 > Thread2> Thread3> Thread4

Highest Lowest

This priority is entire reversed in some systems. This may be due to some config parameter setting in Linux.

Checking Output:

Each thread prints to separate column, thread1 prints on leftmost column, thread2 second leftmost and so on.

Each thread first opens the ht530 driver and writes 50 items from key 1 to 50. Thus 200 items are hashed to the table. Thread 1 items has values from 2 to 100 all even numbers. Thread values from 102 to 200. Thread 3 has values 202 to 300 and Thread 4 has values from 302 to 400. All threads write to same keys hence the subsequent thread which ran will overwrite values previously written by other threads.

After Writing is done “WRITING OVER THREAD X” is printed, X stands for thread number.

This is followed by 50 reads by each thread for key 1 to 50. The contents for each key is printed.

This is followed by deletion of the 50th key item by writing with a data value of 0.

Then the 50th key is again read and we see that the key does not exists.

At the end of all threads the contents of all keys which have a data hashed to it are printed in table dump.

**Part 2:**

A header file “commonData.h” and “bufferStruct.h” is included in both the main.c and ht530\_drv.c. It has the common struct definitions.

Gathering information:

1. Run the command **‘objdump -d -j .data ht530\_drv.ko’** and note down the offset of <test>.

Run the command **‘sudo cat /sys/module/ht530\_drv/sections/.data’.** Add this address (data section relocation address) to the offset of <test>. The resultant address is the ***globalAddress***.

1. Decide the function and offset to run kprobe handlers at, by running the command ‘**objdump -d ht530\_drv.ko’** and calculating offset from the function base address. This is the ***functionOffset.*** Example if <ht\_driver\_write starts> starts at 0x60 and if you want to place probe at 0x99 then ***functionOffset*** is 0x39.
2. Similarly, run ‘**objdump -d ht530\_drv.ko’** to calculate the local variable address offset from the probed function, whose value should be returned by Mprobe driver. This address is the **localOffset.** This offset should be calculated from the ebp. For e.g. we observed the following in the disassembly

00000060 <ht\_driver\_write>:

60: 55 push %ebp

.

.

.

78: 85 d2 test %edx,%edx

7a: c7 45 ec 0a 00 00 00 movl $0xa,-0x14(%ebp)

81: c7 05 00 00 00 00 05 movl $0x5,0x0

As we can see here observe that the variable bkt is being given the value 0xa i.e. 10, this goes to ebp – 14, hence we know the location of variable is ebp – 14 on the stack. Therefore, the local offset is 14, which we subtract from the base pointer obtained from pt\_regs in the kernel.

We chose the line below where value 3 is being written to the local variable for inserting the kprobe.

99: c7 45 ec 03 00 00 00 movl $0x3,-0x14(%ebp).

So we subtract the IP 99 from 60 (beginning of ht\_driver\_write) to get the offset 39 which we use as the function offset.

Similar calculation was done for registering kprobe in ht\_driver\_read.

We have used volatile to avoid compiler optimization on local variable bkt (same name variable present in both ht\_driver\_read & ht\_driver\_write function) and global variable test.

The information given below worked for us (and should mostly work for you too). Only the global variable address would be different, which can be easily calculated using the steps given earlier:

functionName: ht\_driver\_read

functionOffset: 0x51

localOffset: 0x18

functionName: ht\_driver\_write

functionOffset: 0x39

localOffset: 0x14

globalAddress:

offset of <test>: 0x6c

Base Address of .data section: 0xf8622000

Steps to run:

The Makefile does the building of the module, the main program and also inserts(insmod) the module.

1. Run terminal and **cd into the project folder**.
2. To build the part **run** the command ***‘sudo make’*** in the terminal. Sudo is necessary as the makefile also inserts the modules.
3. Run the main file by **‘sudo ./main2’**.
4. Enter the ***functionName*, *functionOffset*, *localOffset*, *globalAddress*** when prompted for, in the console. Hex input is expected.

Checking Output.

The test program main2.c has 2 threads. The first thread asks for user input to register the kprobe and the second thread writes and reads from the ht530 and Mprobe drivers. The second thread starts only after first thread ends.

The second thread first calls write on ht530 driver 25 times which triggers the pre-post handlers 25 times and the buffer is filled. Since the ring buffer is of size 5 the entire buffer will be filled 5 times (all data is over-written after 5 writes). It then reads 10 times from the Mprobe read function. If the probe was inserted in the ht\_driver\_write function you would observe that the trace items are printed from the buffer. The ring buffer is of size 5 and we have called read 10 times hence the first 5 reads will print trace items and the buffer will become empty. The next 5 prints will show an error message for EINVAL as buffer was empty.

If the probe was inserted in the ht\_driver\_read function, then the first 10 prints will read EINVAL message. This is because the kprobe was never triggered and buffer remained empty when the write function of ht530 was called 25 times.

Now we call the read function of ht530 25 times. This will first print the contents of the keys provided in each call. If the krpobe is registered in ht\_driver\_write the subsequent 10 reads from Mprobe driver will contain the EINVAL message as the kprobe is not triggered and buffer remains empty. This proves the kprobe is not triggered from the wrong function.

The thread 1 again starts and asks to register a new Kprobe. User can register the Kprobe now in ht\_driver\_read with the appropriate address and observe that the kprobe is not triggered when write is called 25 times on ht530 driver. Thus the first 10 prints after Mprobe read contain the EINVAL message as the buffer is empty.

Now as the read function of ht530 is called 25 times, kprobe is triggered. The next 10 Mprobe reads print the trace items 5 times and EINVAL message 5 times as the limit of buffer size permits.