**RESULTS**

**PART 1: HAST LIST DRIVER FUNCTIONALITY**

**Testing independent functions**

1. Read and Write Functions (Test\_Read\_Write.c)

In this file, we first print an empty hash table using ioctl functionality. Then, we are writing 3 (key, data) pair into a Hashtable as per our choice and we again print the hashtable using ioctl functionality. Then we test for deleting functionality by writing data = 0 for key = 303. Next we read key = 303 and are unable to find the same in hashlist. Thus cross verifying all read and write cases

2. IOCTL Function (Test\_IOCTL.c)

Here, We have written elements

data = 1, key = 303

data = 2, key = 21

data = 3, key = 70

and then print them

we know that (1,303) and (2, 70) are in the bucket no 94. Thus we use ioctl to print bucket 94.

**Note: The images for the above testing are enclosed /Images-Testing & /Images – Hash – Part1 folder**

**PART2: KPROBE: HAST LIST DRIVER FUNCTIONALITY**

It will print the following data:

Data captured by kprobe, Kprobe address, Timestamp , PID and local variable value

printk(KERN\_INFO "Mprobe - Data captured by Kprobe\n");

printk(KERN\_INFO "Mprobe - Kprobe address: %p\n", probe\_data.kprobe\_addr);

printk(KERN\_INFO "Mprobe - Timestamp of probe: %lu\n", probe\_data.tsc);

printk(KERN\_INFO "Mprobe - Process pid: %d\n",probe\_data.pid);

printk(KERN\_INFO "Mprobe - Local variable value: %d\n",probe\_data.value);

User Inputs the offset address of the hash\_driver write function where the probe is to be inserted. When there is a probe hit, the pre and post handler method of kprobe print the above data in kernel. We use the pre\_handler method to store the data in a circular buffer

**Output**

In this first part we are expected to have 5 threads that simultaneously run to fill the two hash tables randomly for maximum of 200 times (MAX\_HASHTABLE\_DATA) and then randomly delete, write and read from the tables for a maximum of 100 times (MAX\_TABLE\_OPERATION).

Upon Completion we are expected to use IOCTL to dump the values of both the hash tables as per IOCTL command.

Operations are printed out as:

WRITE → writing a value pair to table → Output reads back to check if its written succesfully

DELET → deleting object if data = 0 → The printout shows data not found in the output showing successful deletion

READI → printout the pair of data if found; else shows not found

**FUNCTION LIST**

**Hash\_Driver**

ht530\_drv\_driver\_open →

ht530\_drv\_driver\_write →

ht530\_drv\_driver\_read →

ht530\_drv\_driver\_release →

ht530\_drv\_driver\_ioctl →

ht530\_drv\_driver\_init →

ht530\_drv\_driver\_exit →