RESULTS

PART 1: HAST LIST DRIVER FUNCTIONALITY

Testing independent functions

1. Read and Write Functions (Test_Read_Write.c)

Here, we are writing 3 (key, data) pair into a Hashtable as per our choice and read a single element from the earlier pair. Then we test for deleting functionality by writing data = 0 for key = 70. Next we read key = 70 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.

3. Open and Remove Functionality

These output are shown in the 3. xxx image enclosed. Where when ever the port is opened or closed, you can see kernel printout

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 →