CS614-Assignment3-PCI Device Driver

1- Design

Design consists of a library file **crypter.c** and a device driver file **cryptocard_mod.c.** User program can access all the services of the cryptocard device using an interface provided by **crypter.c** which interacts with device driver **cryptocard_mod.c** using character device and sysfs.

Library: crypter.c

- It provides interface to user program to use services of cryptocard.
- Newly created DEV_HANDLE is inserted into linked list hence changing any configuration of one handle do not affect another.
- Encryption/Decryption can be done on any arbitrary data as operation is performed on chunked data.

Driver:cryptocard_mod.c

- Encryption/Decryption can be performed on writing to character device and reading the result back from it.sysfs is used for it.
- Concurrency is supported with the help of mutex_list.Operation is performed by acquiring lock(If available) from mutex_list,set configuration,perform and then release lock.
- Configuration can be set using sysfs variables:INTERUPT,DMA,KEY_A,KEY_B,IS_MAPPED

Implementation

- **Library**:crypter.c
  - ☐ Creates new handle by calling **create_handle()** .It adds newly created handler to linkedList.The LinkedList contains list of all opened chardev in the head of structure **dev_handle**.Apart from chardev,the structure also contains device configuration information.
  - ☐ There are some utility functions for adding(**_add_handle()**),deleting(**_remove_handle()**)and fetching(**_get_handle()**) dev_handle of a chardev.
  - ☐ To get hold of driver**,_lock_device()** function is used.It get a hold by writing its tid to **sysfs TID**.**_unlock_device()** writes -1 to **sysfs TID**.
  - ☐ The **_set_device_config()** is called for setting the device configuration.

- [ ] The **encrypt()** function takes the hold of lock using **_lock_device().**It then sets the device configuration using **_set_device_config()**.It thus calls **_device_operate()** to operate on data in different chunks,then calls **_unlock_device()** to release the device.
- [ ] The **decrypt()** follows same steps as of **encrypt().**

- [ ] Macros like MMIO_BATCH_SIZE and DMA_BATCH_SIZE is used to determine the chunk size for mmio and dma respectively.It maximum data size on which device can operate on selected mode.
- [ ] **set_key()** add key_A and key_B data to struct dev_handle.
- [ ] **map_card()** and **unmap_card()** shifts the pointer with DEVICE_MEM_OFFSET which basically points to the start of unused memory region where device expects data.

- **Driver: cryptocard.c**
  - [ ] It consists of function to setup PCI device and character device.PCI device information is added in the structure **my_driver_id_table[].**struct **my_driver** contains structures and functions to setup pci device.
  - [ ] All the operations on character device is mentioned in **struct fops**
  - [ ] All sysfs variables except TID,are in an attribute group controlled by **sysfs_store()** and **sysfs_show().**
  - [ ] **sysfs_store_tid()** and **sysfs_show_tid()** controls sysfs varialble TID. TID at any time is either the tid of the task currently operating the device or -1 if free. **sysfs_store()** is protected by a mutex dev_lock. Each task tries to acquire dev_lock, then writes its tid to TID. Now the task is free to operates on the device. Once done, this task has to store -1 to TID which is accompanied by releasing dev_lock.
  - [ ] **cdev_write()** is characted device write function. If memory is not mapped, it first copies data to mmio_buf/dma_buf and then calls **mmio()/dma()**. This functions returns only when result is available.
  - [ ] **cdev_read()** is called for reading result. The result is guranteed to be available when called.
  - [ ] Wrapper function **write_to_device()** and **read_to_device()** are used for implement copy_to_user and copy_from_user.
  - [ ] **mmio()** and **dma()** either polls the status register in case of non-interrupt mode, else in interruptv mode, it waits on a wait queue (wq_is_data_ready) in intereruptable mode.
  - [ ] **irq_handler()** is called in interrupt mode. It signals the wq_is_data_ready wait queue.

- **Testing:**
  - ☐ Tested and passed testcases given for each operation in benchmark-test.
  - ☐ Modified the test1.c file to test whether device is able to handle large data to read and write.More specific to chunk creation.
  - ☐ Tested over multi-process environment.

- **Benchmark:**

  Collect stat every 2 seconds on 5MB file using **“sar -u 2 20”**,presenting the average result.

Program	**%user**	**%nice**	**%system**	**%iowait**	**%steal**
mmio	0.10	0.00	51.88	0.29	0.08
mmio_int	1.21	0.00	43.10	4.20	0.09
dma	0.4	0.00	50.68	1.38	0.04
dma_int	0.49	0.00	0.83	1.17	0.05
mmap	3.65	0.00	24.42	1.97	0.08
mmap_int	4.05	0.00	26.35	0.52	0.09