Operating Systems

Assignment 2

"Synchronization in Linux Kernel"

Name: Ahmed Tarek Ibrahim ElGamal
ID: 09

How is the code Designed?

- Kernel Module: Module code is divided into two parts:
    - first part : Contacting with the keyboard
    - second part : Interfacing User-Space Programs.
    - In addition to module initialization part.

First Part: contains set_led_state, get_led_state, update_led_status, kbd_read_status, kbd_read_data, kbd_write_data fuctions

When value inside the file is updated, show functions (within the 2nd part) calls set_led_state( ) giving it the value in the file, then set_led_state( ) prepares status word and calls update_leds( ) with status word as argument.

Update_leds( ) communicates with keyboard giving it the new status word.

As general, second part code works as listener on sys files and calls set_led_state function from first part to update the LEDs state which calls update_leds ( ) that communicates with keyboard through kbd_read_status( ), kbd_read_ data( ) and kbd_write_data( )

- As for race conditions and semaphores, semaphores are set on set_led_state ( ) function as any race condition would happen inside it. As get_led_state does not have any variable assignment.

What are the main functions?

- update_leds(unsigned char status_word) :
notifies the keyboard for set LEDs command, waits for acknowledgement, then send LED states (status word) and waits for acknowledgement from keyboard.

- set_led_state(int led, int state) :
takes led index and state as args, prepares the status word and calls update_leds giving it the status word. It is being called by _show functions (that listens on the sys files).

- get_led_state(int led) :
returns the state of the led mentioned at the arguments, whether it is on or off.

- kbd_read_status( ) :
returns the status byte to the caller.

- kbd_read_data( ) :

- reads STATUS register using kbd read status().
- checks bit 0 of the STATUS register. If bit 0 is clear, this means that PS/2 controller didn’t receive anything from the keyboard up till the moment. Goto step 1 and repeat the process again. Otherwise (bit 0 is set), continue to step 3.
- if control reaches this step, this means that the keyboard has actually sent some data to keyboard controller. Read the data using inb(0x60).
- returns data to the caller.

-kbd_write_data( ) :

- reads STATUS register using kbd read status().
- checks bit 1 of the STATUS register. If bit 1 is set, this means that PS/2 controller is still processing last command/data sent to it. Goto step 1. Otherwise (bit 1 is clear), continue to step 3.
- if control reaches this step, this means that the keyboard controller is ready to receive data into its input buffer. Write the data using outb(data, 0x60). The 8-bit data value will then be sent to PS/2 keyboard

- (num/scroll/caps)_show ( ) :
automatically reads the sys files associated with the LED when updated, stores it into buffer.

- (num/scroll/caps)_store ( ) :
automatically discovers that the sys files associated with the LED has been updated, updates the static variable representing the state at the file and calls set_led_state function to update the led state.

- module_init ( ) :
runs at module loading, initializes the module as it initializes the semaphore and create subdirectory for sys files in /sys/kernel.

- module_exit ( ) :
runs at module removing from memory, frees the subdirectory created for sys files.

How to compile and run the code?

- Kernel Module :

- compiled through Makefile sent with report.
- Loaded into memory through terminal command
  sudo insmod module_name.ko
- Unloaded from memory through:
  sudo rmmod module_name

- Interfacing Unit (leds):

- can be compiled through Makefile sent with the report or through terminal :
  gcc -o [[exe_name]] [[src_code.c]]

- can be called through commands :
  sudo ./leds set caps on
  sudo ./leds set num off
  (sudo) ./leds get num

Sample Runs:

simple set and get

```
ahmedelgamal@ahmedelgamal:~$ cd Desktop/controller/
ahmedelgamal@ahmedelgamal:~/Desktop/controller$ gcc -o leds main.c
ahmedelgamal@ahmedelgamal:~/Desktop/controller$ sudo ./leds get caps
[sudo] password for ahmedelgamal:
on
ahmedelgamal@ahmedelgamal:~/Desktop/controller$ sudo ./leds set caps off
ahmedelgamal@ahmedelgamal:~/Desktop/controller$ sudo ./leds get caps
off
ahmedelgamal@ahmedelgamal:~/Desktop/controller$ sudo ./leds set num on
ahmedelgamal@ahmedelgamal:~/Desktop/controller$ sudo ./leds get num
on
ahmedelgamal@ahmedelgamal:~/Desktop/controller$
```

Before Enabling Semaphores :

```
[33003.483649] PID: 22007 || START.
[33003.483658] PID: 22007 || SENDING 0xED COMMAND.
[33003.483944] PID: 22007 || RECEIVED ACK.
[33003.483946] PID: 22007 || SLEEP.
[33003.486231] PID: 22008 || START.
[33003.486240] PID: 22008 || SENDING 0xED COMMAND.
[33003.487256] PID: 22008 || RECEIVED ACK.
[33003.487259] PID: 22008 || SLEEP.
[33003.986123] PID: 22007 || WAKE UP.
[33003.986131] PID: 22007 || SENDING KEYBOARD DATA.
[33003.990124] PID: 22008 || WAKE UP.
[33003.990134] PID: 22008 || SENDING KEYBOARD DATA.
```

after adding semaphores