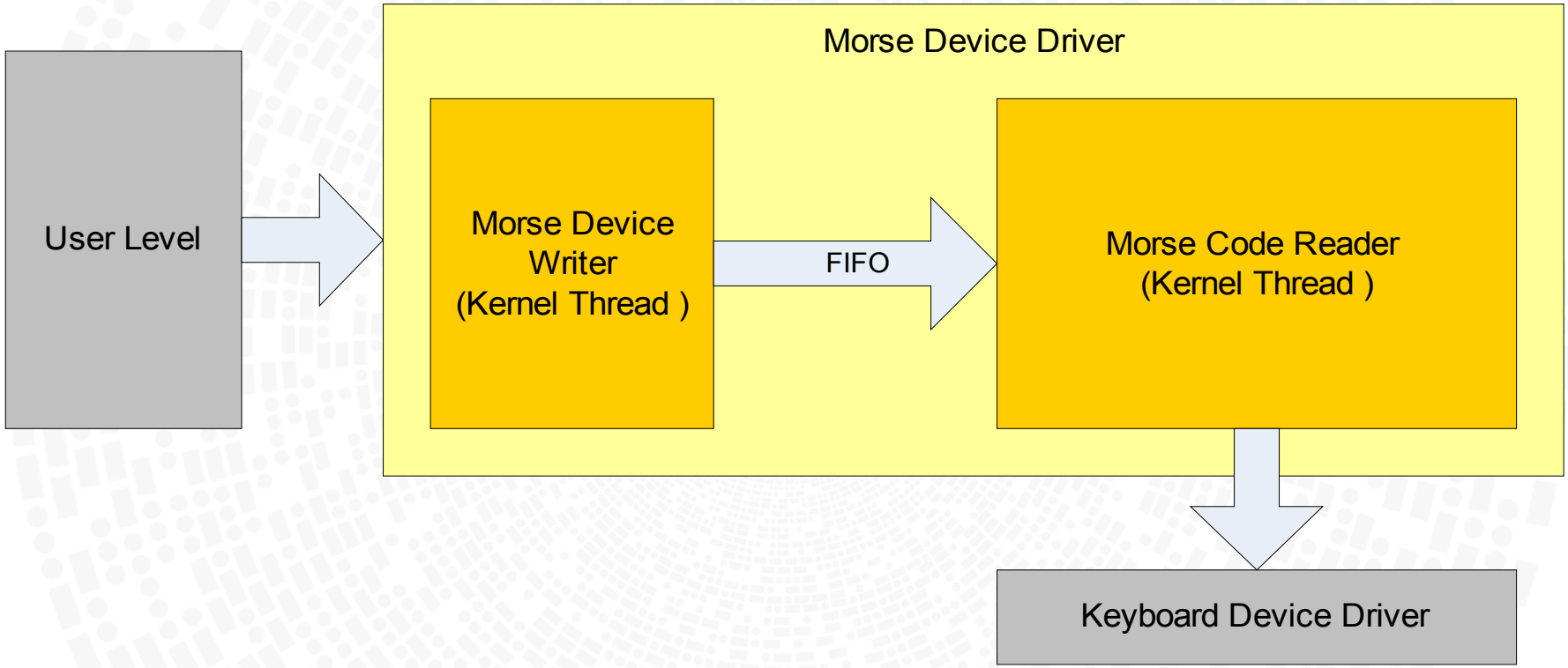
Operation System Project

Linux device driver for morse code

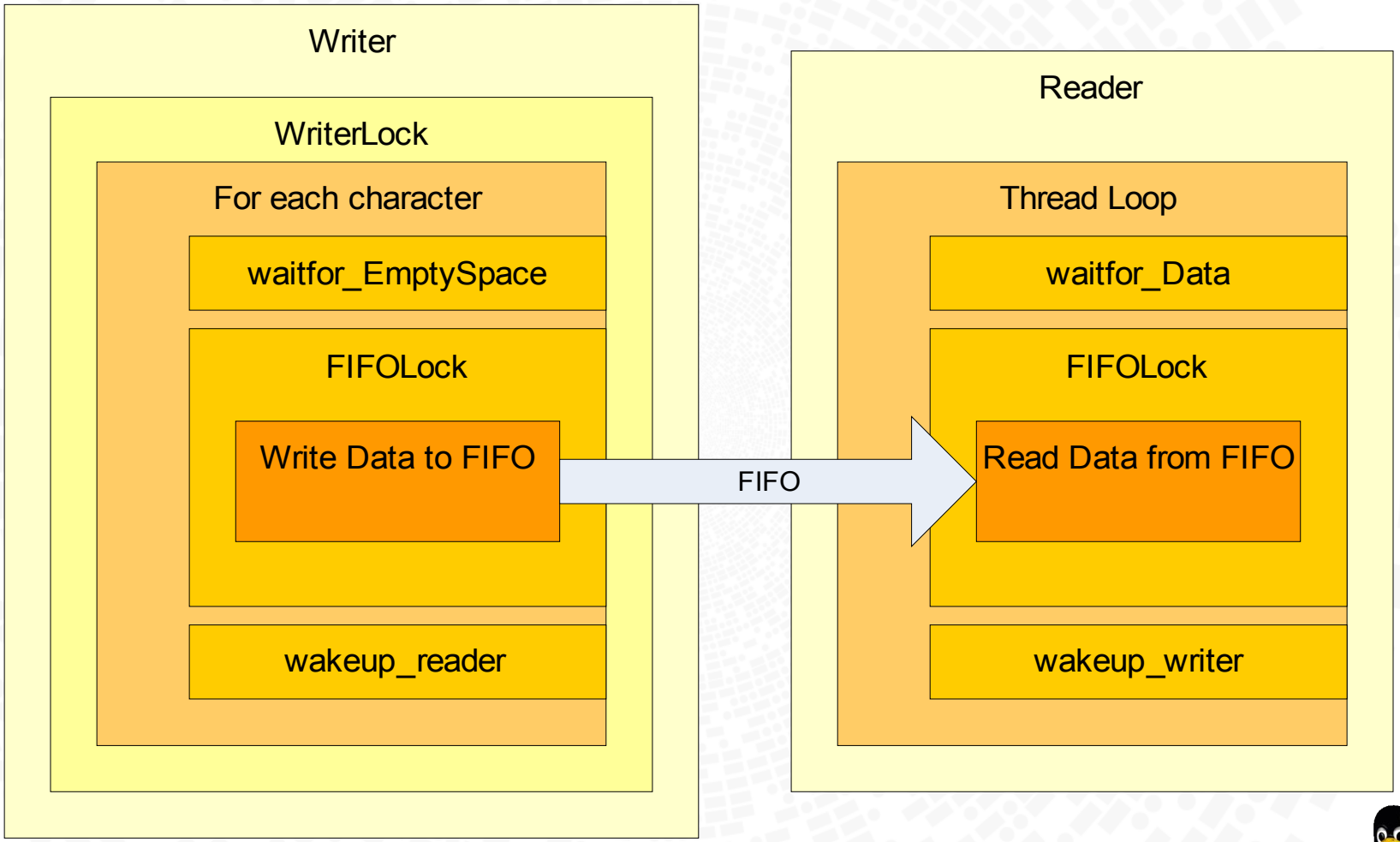
Overview

- Design
- Implementation
  - Writer
  - Reader
- Conclusion

Design (I)

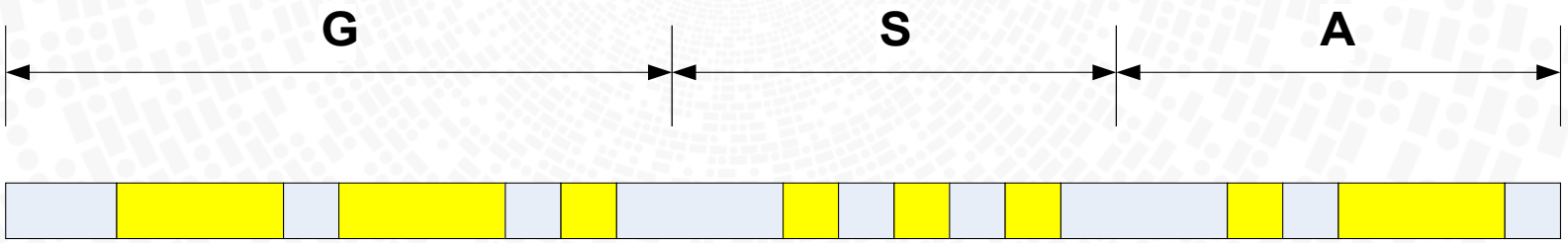

Morse Device Driver
User Level
Morse Device Writer (Kernel Thread)
FIFO
Morse Code Reader (Kernel Thread)
Keyboard Device Driver

Design (II)


Writer
WriterLock
For each character
waitfor_EmptySpace
FIFOLock
Write Data to FIFO
wakeup_reader
FIFO
Reader
Thread Loop
waitfor_Data
FIFOLock
Read Data from FIFO
wakeup_writer

Design (III)

- The morse code is stored as binary strings inside a table
- The original character is used as table index
- Only one time slice is used to generate a morse code
  (**3x for dashes**; **1x for dots**; 1x break signs; 3x break chars)



Implementation (I)

```
struct TypeMorseDevice
  {
    struct cdev           cdev;
    struct task_struct*   pThread;
    struct kfifo*         pFifo;
    struct semaphore      lockMutex;
    struct semaphore      lockWriter;
    char*                 pcFifoBuffer;
    struct tty_driver*    pTTYDriver;
  };
```

- Device structure used for
  - Communication objects
  - device objects

Implementation (II)

```
static struct file_operations g_Main_fops =
    {
        .write   = main_write,
        .open    = main_open,
        .release = main_release,
        .owner   = THIS_MODULE,
    };
```

- The file operation structure
  - Opens and releases a connection to the FIFO (virtual only)
  - writes data to the FIFO queue

Writing data

- Only one writer can access the FIFO
- Writing data is done in 4 steps
  - 1. check if FIFO is full -> writer is suspended
  - 2. acquire the FIFO lock
  - 3. write data to the FIFO and release lock
  - 4. wake up the possible waiting reader
  - step 1 to 4 are repeated until all data is written

Reading data

- Reading is in 5 steps
  - 1. check if FIFO is empty -> reader is suspended
  - 2. acquire the FIFO lock
  - 3. read data from FIFO into local variable and release the FIFO lock
  - 4. wake up the possible waiting writer
  - 5. send the data as morse code

Parameters

- Parameters of the driver
  - BlinkTime: Time slice of the morse code
  - BufferSize: Size of the buffer as exponent of 2
    5 = 2^5 = 32 Bytes
  - Mo_Major: Major device number of the driver.
- Parameters can be set using the install script

```
# ./Install.sh BufferSize=10 BlinkTime=75
```

Conclusion

- The project gives a very good idea how Linux driver work
  - Some simple rules to meet the needs of a character device driver
- Very interesting project work
- How about Windows Drivers?