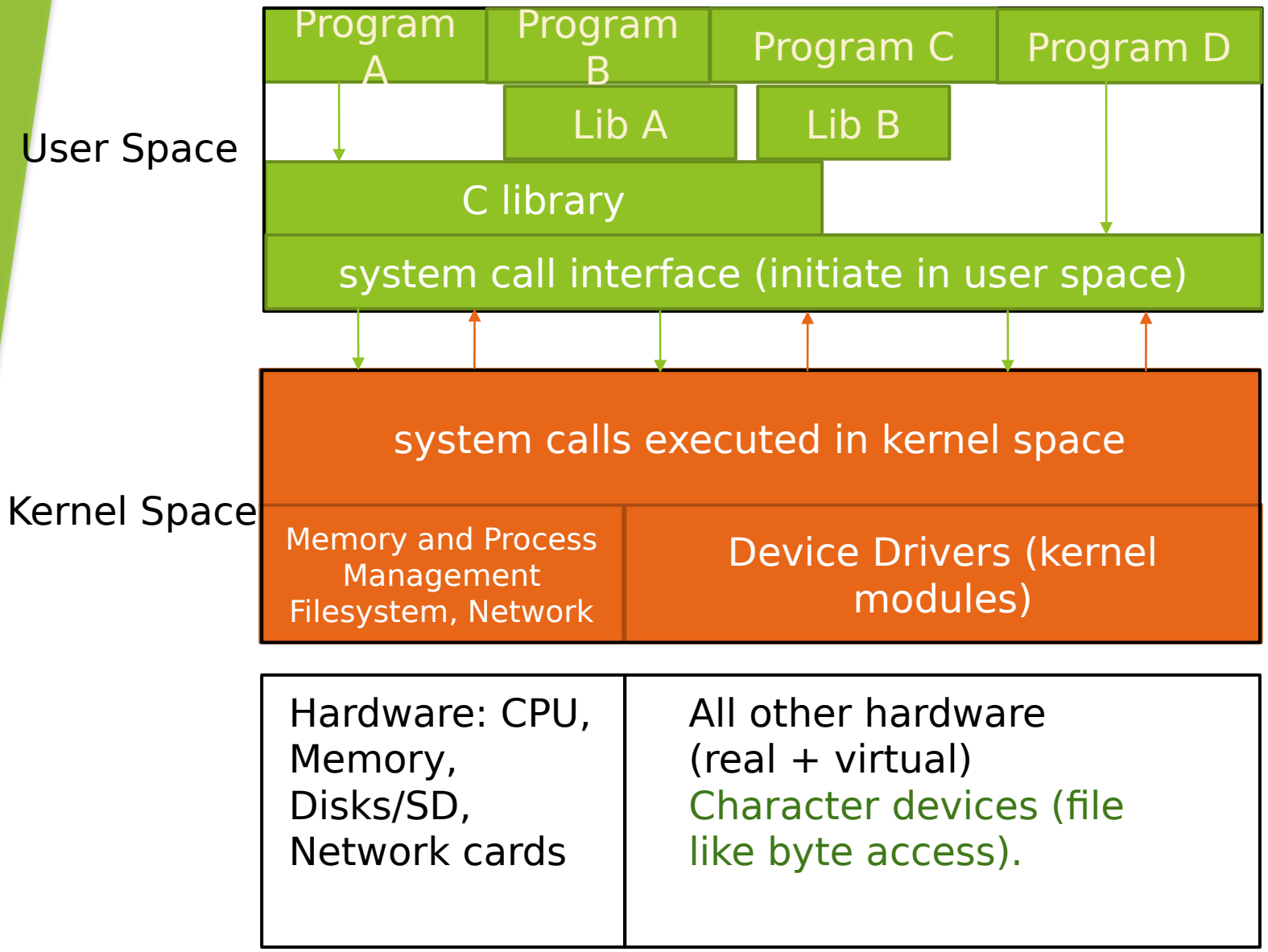
# Operating Systems

Kernel Modules

# Kernel Space vs User Space




C standard lib
System calls

no c standard lib

# Kernel Modules

- Linux executable and linkable format (ELF)
  - Use gcc to compile/cross-compile
  - Runs in the kernel address space (high privilege)
  - Have use the extension “.ko”
- Loaded dynamically
  - Virtual
  - Control for some hardware device
- Manual load
  - insmod (will call the init callback)
    - Reserve resources, register callbacks
    - Create a “device file”/inode structure that includes a “file_operations” struct
  - rmmod (will call the exit callback)
    - Release resources (IRQ’s, locks), free memory
- No main() entry point
  - init() and exit() points / callback functions
    - More callback functions can be added in the init() routine

# Character Device

5bfc75d92e linux / include / linux / cdev.h

gregkh License cleanup: add SPDX GPL-2.0 license identifier to files with no...

8 contributors

39 lines (28 sloc) | 845 Bytes

```
/* SPDX-License-Identifier: GPL-2.0 */
#ifndef _LINUX_CDEV_H
#define _LINUX_CDEV_H

#include <linux/kobject.h>
#include <linux/kdev_t.h>
#include <linux/list.h>
#include <linux/device.h>

struct file_operations;
struct inode;
struct module;

struct cdev {
        struct kobject kobj;
        struct module *owner;
        const struct file_operations *ops;
        struct list_head list;
        dev_t dev;
        unsigned int count;
} __randomize_layout;

```

*Source: https://github.com/torvalds/linux/blob/5bfc75d92efd494db37f5c4c173d3639d4772966/include/linux/cdev.h*

# File Operations

```
struct file_operations {
        struct module *owner;
         loff_t(*llseek) (struct file *, loff_t, int);
         ssize_t(*read) (struct file *, char __user *, size_t, loff_t *);
         ssize_t(*aio_read) (struct kiocb *, char __user *, size_t, loff_t);
         ssize_t(*write) (struct file *, const char __user *, size_t, loff_t *);
         ssize_t(*aio_write) (struct kiocb *, const char __user *, size_t,
                              loff_t);
        int (*readdir) (struct file *, void *, filldir_t);
        unsigned int (*poll) (struct file *, struct poll_table_struct *);
        int (*ioctl) (struct inode *, struct file *, unsigned int,
                      unsigned long);
        int (*mmap) (struct file *, struct vm_area_struct *);
        int (*open) (struct inode *, struct file *);
        int (*flush) (struct file *);
        int (*release) (struct inode *, struct file *);
        int (*fsync) (struct file *, struct dentry *, int datasync);
        int (*aio_fsync) (struct kiocb *, int datasync);
        int (*fasync) (int, struct file *, int);
        int (*lock) (struct file *, int, struct file_lock *);
         ssize_t(*readv) (struct file *, const struct iovec *, unsigned long,
                          loff_t *);
         ssize_t(*writev) (struct file *, const struct iovec *, unsigned long,
                           loff_t *);
         ssize_t(*sendfile) (struct file *, loff_t *, size_t, read_actor_t,
                             void __user *);
         ssize_t(*sendpage) (struct file *, struct page *, int, size_t,
                             loff_t *, int);
        unsigned long (*get_unmapped_area) (struct file *, unsigned long,
                                            unsigned long, unsigned long,
                                            unsigned long);
};
```

*Source: https://tldp.org/LDP/lkmpg/2.6/html/x569.html*

# Basic Module

Find your version number using: uname –a
ll /usr/src

test_004_pthreads.c | test_006_pthreads_cond.c | Makefile | test_mod00.c 4

test_mod00.c > MODULE_LICENSE

```c
#include <linux/module.h>
#include <linux/kernel.h>
#include <linux/kern_levels.h>

MODULE_LICENSE("GPL");

int ece_init(void)
{
    printk(KERN_INFO " ECE4310: Start here \n");
    return 0;
}

void ece_end(void)
{
    printk(KERN_INFO " ECE4310: End here \n");
}

module_init(ece_init);
module_exit(ece_end);
```

Makefile | test_mod00.c 4 | test_mod01.c | read_write_mod.c

Makefile

```makefile
obj-m += test_mod00.o

KDIR=/usr/src/linux-headers-5.11.0-38-generic

CFLAGS = -D__KERNEL__ -DMODULE -I$(KDIR)/include -Wall

all:
    $(MAKE) -C $(KDIR) M=$(shell pwd) $(KCONFIG) modules

clean:
    $(MAKE) -C $(KDIR) M=$(shell pwd) clean
    rm -f *.o
```

*Linux man-pages: full reference, not for learning. Check here when you know what you are looking for. (https://www.kernel.org/doc/man-pages/)*

# Basic Module

```
loniciuc@loniciuc-UVM:~$ ll /usr/src/
total 32
drwxr-xr-x  8 root root 4096 Nov 15 19:31 ./
drwxr-xr-x 14 root root 4096 Feb  9  2021 ../
drwxr-xr-x  7 root root 4096 Sep 29 18:53 linux-headers-5.11.0-37-generic/
drwxr-xr-x  7 root root 4096 Oct 20 06:21 linux-headers-5.11.0-38-generic/
drwxr-xr-x  7 root root 4096 Nov 15 19:31 linux-headers-5.11.0-40-generic/
drwxr-xr-x 24 root root 4096 Sep 29 18:53 linux-hwe-5.11-headers-5.11.0-37/
drwxr-xr-x 24 root root 4096 Oct 20 06:21 linux-hwe-5.11-headers-5.11.0-38/
drwxr-xr-x 24 root root 4096 Nov 15 19:30 linux-hwe-5.11-headers-5.11.0-40/
loniciuc@loniciuc-UVM:~$ uname -a
Linux loniciuc-UVM 5.11.0-38-generic #42~20.04.1-Ubuntu SMP Tue Sep 28 20:41:07 U
_64 x86_64 x86_64 GNU/Linux
```

- **make**
- *lsmod*
- *lsmod | grep test*
- **sudo insmod test_mod00.ko**
- *lsmod | grep test*
- *dmesg | tail*
- ***sudo rmmod test_mod00***
- *dmesg | tail*

*Linux man-pages: full reference, not for learning. Check here when you know what you are looking for. (https://www.kernel.org/doc/man-pages/)*

# Globals

```
MODULE_LICENSE("GPL");

#define ECE_BUF_SIZE 4096

static char ece_buffer[ECE_BUF_SIZE];
int isReg;
int major;
int ece_offset_w;
int ece_offset_r;
int ece_size;
```

# Write

```c
static ssize_t ece_write(struct file *fp, const char *buf, size_t count, loff_t *op)
{
    int ret = 0;
    if(ece_offset_w + count >= ECE_BUF_SIZE)
    {
        printk(KERN_INFO " ECE4310: Write OverFlow. Abort. \n");
        return -1;
    }

    printk(KERN_INFO " ECE4310: Copy from user. \n");
    ret = copy_from_user(&ece_buffer[ece_offset_w], buf, count);
    if(ret != 0)
    {
        printk(KERN_INFO " ECE4310: ERR copy from user. \n");
        return -1;
    }
    ece_offset_w = ece_offset_w + count;

    return count;
}
```

# Read

```c
static ssize_t ece_read(struct file *fp,
    char *buf,        /* to fill with data for user */
    size_t count,     /* how much data to send      */
    loff_t *offset)   /* take the data from offset  */
{
    int ret;
    if(ece_offset_r + count >= ECE_BUF_SIZE)
    {
        printk(KERN_INFO " ECE4310: Read OverFlow. Abort. \n");
        return -1;
    }
    printk(KERN_INFO " ECE4310: Copy to user. \n");
    ret = copy_to_user(buf, &ece_buffer[ece_offset_r], count);
    if(ret != 0)
    {
        printk(KERN_INFO " ECE4310: ERR copy to user. \n");
        return -1;
    }

    ece_offset_r = ece_offset_r + count;

    return count;
}
```

# Updated init

```
static struct file_operations ece_fops =
{
    .read = ece_read,
    .write = ece_write,
};

int ece_init(void)
{
    int ret = 0;
    major = register_chrdev(0, "test_mod01", &ece_fops);
    ece_offset_w = 0;
    ece_offset_r = 0;
    ece_size = 0;

    if(major < 0)
    {
        isReg = 0;
        printk(KERN_INFO " ECE4310: Start FAIL \n");
    }
    else
    {
        isReg = 1;
        printk(KERN_INFO " ECE4310: Start here \n");
    }
    return 0;
}
```

# Updated end function

```
void ece_end(void)
{
    if(isReg)  //if(major >= 0)
    {
        unregister_chrdev(major, "test_mod01");
    }
    printk(KERN_INFO " ECE4310: End here \n");
}

module_init(ece_init);
module_exit(ece_end);
```

# Kernel Modules

- Major/Minor Numbers

```
loniciuc@loniciuc-UVM:c$ cat /proc/devices | grep test
237 test_mod01
```

```
loniciuc@loniciuc-UVM:c$ sudo mknod /dev/test_mod01 c 237 1
```

```
loniciuc@loniciuc-UVM:c$ ll /dev/ | grep test
crw-r--r--   1 root     root    237,   1 Nov 14 19:18 test_mod01
```

```
loniciuc@loniciuc-UVM:c$ sudo chmod 0777 /dev/test_mod01
loniciuc@loniciuc-UVM:c$ ll /dev/test_mod01
crwxrwxrwx 1 root root 237, 1 Nov 14 19:18 /dev/test_mod01
```

# Kernel Modules

```
loniciuc@loniciuc-UVM:c$ echo "HI CLASS.... is it late?" > /dev/test_mod01
loniciuc@loniciuc-UVM:c$ echo "maybe...." > /dev/test_mod01
loniciuc@loniciuc-UVM:c$ echo "SOME LOG MESSAGE IN HERE" > /dev/test_mod01
loniciuc@loniciuc-UVM:c$ ./a.out


result: First TestS▒

loniciuc@loniciuc-UVM:c$ ./a.out


result: New messag▒

loniciuc@loniciuc-UVM:c$ ./a.out


result: HI CLASS..▒▒

loniciuc@loniciuc-UVM:c$ ./a.out


result: .. is it l3

loniciuc@loniciuc-UVM:c$ ./a.out


result: ate?
maybeR

loniciuc@loniciuc-UVM:c$
```

# Kernel Modules

- *lsmod | grep test*
- *sudo chmod +777 /dev/test_mod01*

- **sudo ./a.out**

| Command | Description |
|---|---|
| **gcc read_write_mod.c** | compile user space application |
| make | compile kernel module |
| *lsmod | grep "test"* | List modules, filter using string "test" |
| **sudo insmod test_mod01.ko** | Insert new compiled module to our kernel |
| | |
| | |
| | |

# Resources

- The Linux Kernel Module Programming Guide
  - https://tldp.org/LDP/lkmpg/2.6/html/
- Linux Device Drivers
  - https://lwn.net/Kernel/LDD3/
- Linux Kernel Sources
  - https://github.com/torvalds/linux
  - https://www.kernel.org/