Operating systems fundamentals - B11

David Kendall

Northumbria University

Outline

- Device classes
- Kernel interface
- Loadable modules
- Useful commands

Introduction

Device drivers implement the *hardware abstraction layer* between software and hardware

- Provide a standard interface to devices, hiding the details of how they work
- Most system operations eventually end in some interaction with a device
- OS kernel contains device drivers for all devices in the system
- Difficult to write device drivers
- Code is often buggy

Device classes

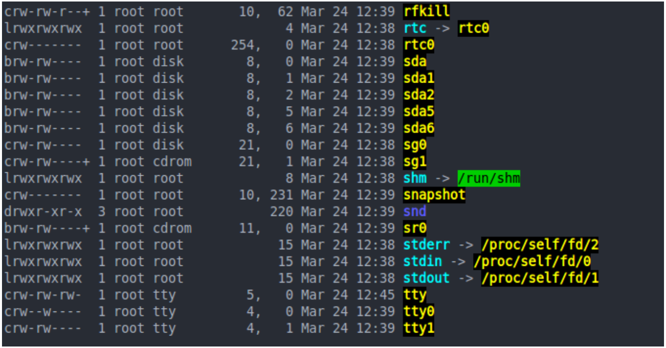
Typically, devices are classified as

- *Character devices*: the simplest devices; accessed as a stream of bytes, sequentially
- *Block devices*: I/O operations transfer whole blocks (e.g. 512 bytes), can be accessed by block address (index) and host a file system
- *Network devices*: packet delivery, use network subsystem; do not use device nodes for access

But this classification is not universal, e.g. a USB device might appear as a character device (serial port), a block device (e.g. flash drive) or network device (e.g. USB Ethernet interface)

Device classes

```
ls -l /dev/
```



```
crw-rw-r--+ 1 root root      10,  62 Mar 24 12:39 rfkill
lrwxrwxrwx  1 root root            4 Mar 24 12:38 rtc -> rtc0
crw-------  1 root root     254,   0 Mar 24 12:38 rtc0
brw-rw----  1 root disk       8,   0 Mar 24 12:39 sda
brw-rw----  1 root disk       8,   1 Mar 24 12:39 sda1
brw-rw----  1 root disk       8,   2 Mar 24 12:39 sda2
brw-rw----  1 root disk       8,   5 Mar 24 12:39 sda5
brw-rw----  1 root disk       8,   6 Mar 24 12:39 sda6
crw-rw----  1 root disk      21,   0 Mar 24 12:38 sg0
crw-rw----+ 1 root cdrom     21,   1 Mar 24 12:38 sg1
lrwxrwxrwx  1 root root            8 Mar 24 12:38 shm -> /run/shm
crw-------  1 root root      10, 231 Mar 24 12:39 snapshot
drwxr-xr-x  3 root root          220 Mar 24 12:39 snd
brw-rw----+ 1 root cdrom     11,   0 Mar 24 12:39 sr0
lrwxrwxrwx  1 root root           15 Mar 24 12:38 stderr -> /proc/self/fd/2
lrwxrwxrwx  1 root root           15 Mar 24 12:38 stdin -> /proc/self/fd/0
lrwxrwxrwx  1 root root           15 Mar 24 12:38 stdout -> /proc/self/fd/1
crw-rw-rw-  1 root tty        5,   0 Mar 24 12:45 tty
crw--w----  1 root tty        4,   0 Mar 24 12:39 tty0
crw-rw----  1 root tty        4,   1 Mar 24 12:39 tty1
```

- Character devices indicated with a `c` in the leftmost column, e.g.

```
crw-rw----  1 root tty         4,   1 Mar 24 12:39 tty1
```

- Block devices indicated with `b` in the leftmost column, e.g.

```
brw-rw----  1 root disk        8,   1 Mar 24 12:39 sda1
```

Communicating with the kernel

Linux provides different mechanisms for communicating between user and kernel space:

- System calls: functions which are useful for many application (about 380)
- `ioctl` syscall: catch-all for non-standard file operations
- Virtual file systems: procfs, sysfs, configfs, debugfs
- `sysctl`: configure kernel parameters at runtime
- Netlink: sockets-based interface e.g. iptables/netfilter
- Upcall: allows kernel modules start a program in userspace

Everything is a file

- The UNIX philosophy is quoted as "everything is a file". What that really means is that everything is a stream of bytes – in the file system namespace.
- Sockets and pipes are the exception, lacking a file name. A more precise definition would be "everything is a file descriptor"
- The advantage of this approach is that you can use the same (file-based) interfaces on different things.

Accessing devices

From the previous list of interfaces, device drivers usually:

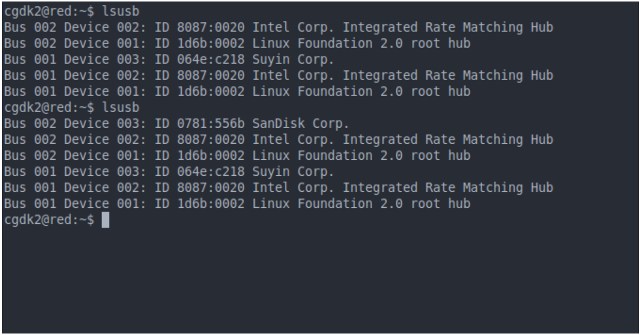
- implement file operations, like `open`, `read`, `write`, `poll`, `mmap` etc. which implement the system calls with the same name
- might use `ioctl` for other device-specific operations, e.g. query capabilities, tuning, etc.
- use the pseudo file system `/proc` to expose internal data structure

Loadable modules

- Linux has the ability to extend the kernel functionality at runtime using modules
- Device drivers can also be added to the kernel in this fashion (benefits are a smaller kernel, on demand loading gives you a better footprint and no kernel recompilation to add new modules)
- You can use the module-init-tools package, which contains a set of programs for loading, inserting and removing kernel modules.

Useful Linux commands for devices

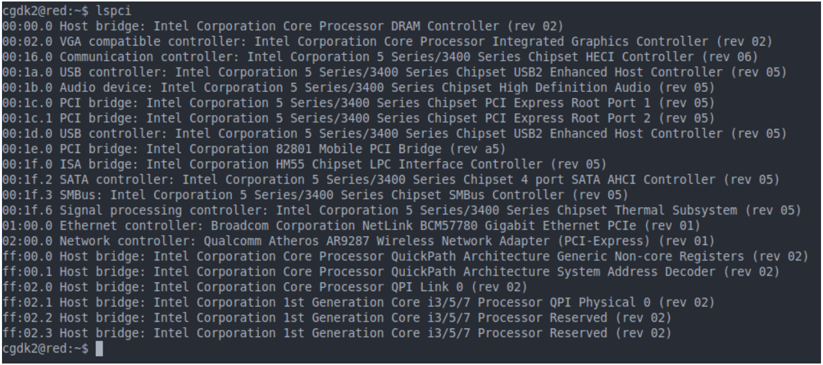
`lsusb`



```
cgdk2@red:~$ lsusb
Bus 002 Device 002: ID 8087:0020 Intel Corp. Integrated Rate Matching Hub
Bus 002 Device 001: ID 1d6b:0002 Linux Foundation 2.0 root hub
Bus 001 Device 003: ID 064e:c218 Suyin Corp.
Bus 001 Device 002: ID 8087:0020 Intel Corp. Integrated Rate Matching Hub
Bus 001 Device 001: ID 1d6b:0002 Linux Foundation 2.0 root hub
cgdk2@red:~$ lsusb
Bus 002 Device 003: ID 0781:556b SanDisk Corp.
Bus 002 Device 002: ID 8087:0020 Intel Corp. Integrated Rate Matching Hub
Bus 002 Device 001: ID 1d6b:0002 Linux Foundation 2.0 root hub
Bus 001 Device 003: ID 064e:c218 Suyin Corp.
Bus 001 Device 002: ID 8087:0020 Intel Corp. Integrated Rate Matching Hub
Bus 001 Device 001: ID 1d6b:0002 Linux Foundation 2.0 root hub
cgdk2@red:~$
```

Useful Linux commands for devices

`lspci`



```
cgdk2@red:~$ lspci
00:00.0 Host bridge: Intel Corporation Core Processor DRAM Controller (rev 02)
00:02.0 VGA compatible controller: Intel Corporation Core Processor Integrated Graphics Controller (rev 02)
00:16.0 Communication controller: Intel Corporation 5 Series/3400 Series Chipset HECI Controller (rev 06)
00:1a.0 USB controller: Intel Corporation 5 Series/3400 Series Chipset USB2 Enhanced Host Controller (rev 05)
00:1b.0 Audio device: Intel Corporation 5 Series/3400 Series Chipset High Definition Audio (rev 05)
00:1c.0 PCI bridge: Intel Corporation 5 Series/3400 Series Chipset PCI Express Root Port 1 (rev 05)
00:1c.1 PCI bridge: Intel Corporation 5 Series/3400 Series Chipset PCI Express Root Port 2 (rev 05)
00:1d.0 USB controller: Intel Corporation 5 Series/3400 Series Chipset USB2 Enhanced Host Controller (rev 05)
00:1e.0 PCI bridge: Intel Corporation 82801 Mobile PCI Bridge (rev a5)
00:1f.0 ISA bridge: Intel Corporation HM55 Chipset LPC Interface Controller (rev 05)
00:1f.2 SATA controller: Intel Corporation 5 Series/3400 Series Chipset 4 port SATA AHCI Controller (rev 05)
00:1f.3 SMBus: Intel Corporation 5 Series/3400 Series Chipset SMBus Controller (rev 05)
00:1f.6 Signal processing controller: Intel Corporation 5 Series/3400 Series Chipset Thermal Subsystem (rev 05)
01:00.0 Ethernet controller: Broadcom Corporation NetLink BCM57780 Gigabit Ethernet PCIe (rev 01)
02:00.0 Network controller: Qualcomm Atheros AR9287 Wireless Network Adapter (PCI-Express) (rev 01)
ff:00.0 Host bridge: Intel Corporation Core Processor QuickPath Architecture Generic Non-core Registers (rev 02)
ff:00.1 Host bridge: Intel Corporation Core Processor QuickPath Architecture System Address Decoder (rev 02)
ff:02.0 Host bridge: Intel Corporation Core Processor QPI Link 0 (rev 02)
ff:02.1 Host bridge: Intel Corporation 1st Generation Core i3/5/7 Processor QPI Physical 0 (rev 02)
ff:02.2 Host bridge: Intel Corporation 1st Generation Core i3/5/7 Processor Reserved (rev 02)
ff:02.3 Host bridge: Intel Corporation 1st Generation Core i3/5/7 Processor Reserved (rev 02)
cgdk2@red:~$
```