An Introduction to Device Drivers

Sarah Diesburg

COP 5641 / CIS 4930

Introduction

- Device drivers
  - Black boxes to hide details of hardware devices
  - Use standardized calls
    - Independent of the specific driver
  - Main role
    - Map standard calls to device-specific operations
  - Can be developed separately from the rest of the kernel
    - Plugged in at runtime when needed

The Role of the Device Driver

- Implements the *mechanisms* to access the hardware
  - E.g., show a disk as an array of data blocks
- Does not force particular *policies* on the user
  - Examples
    - Who many access the drive
    - Whether the drive is accessed via a file system
    - Whether users may mount file systems on the drive

Policy-Free Drivers

- A common practice
  - Support for synchronous/asynchronous operation
  - Be opened multiple times
  - Exploit the full capabilities of the hardware
- Easier user model
- Easier to write and maintain
- To assist users with policies, release device drivers with user programs

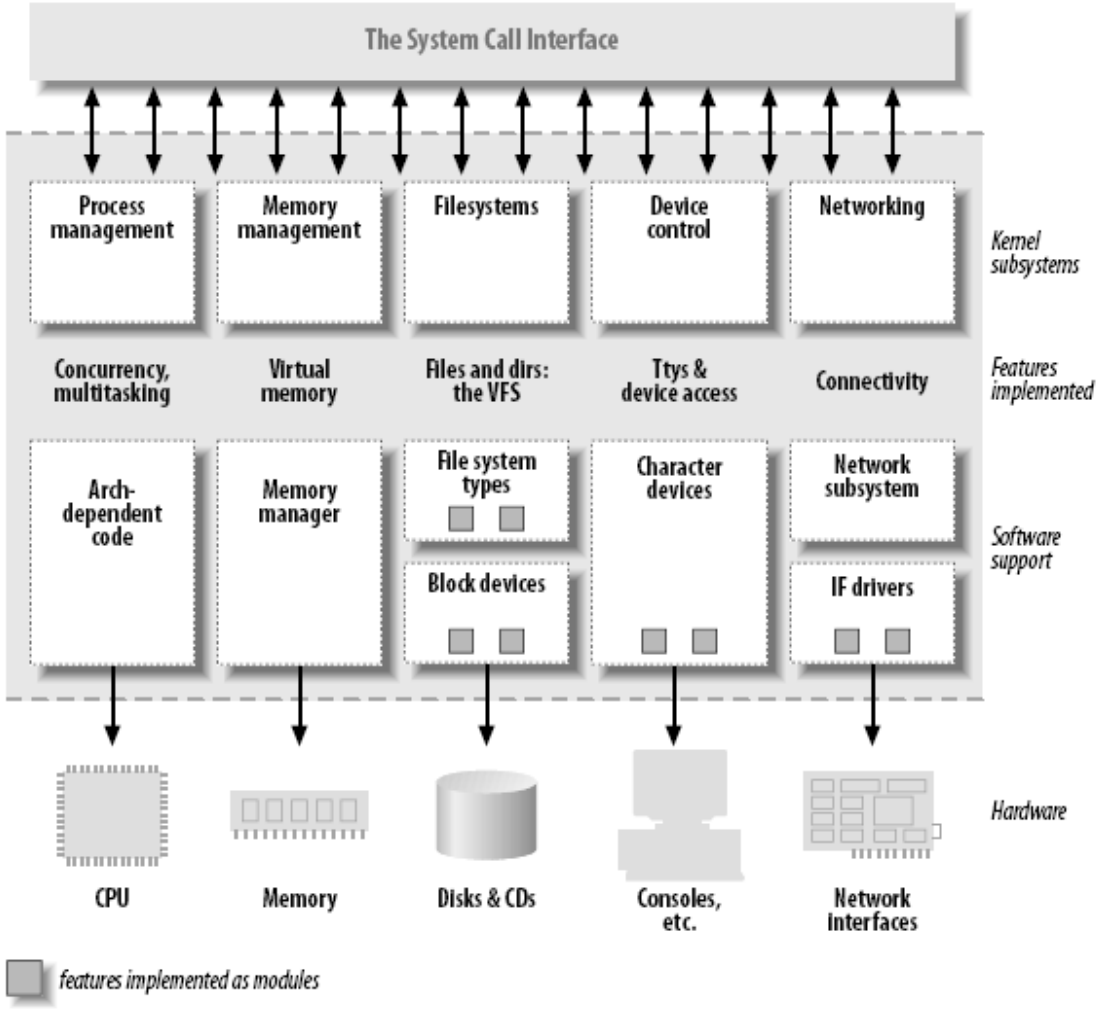
Splitting the Kernel

- Process management
  - Creates, destroys processes
  - Supports communication among processes
    - Signals, pipes, etc.
  - Schedules how processes share the CPU
- Memory management
  - Virtual addressing

Splitting the Kernel

- File systems
  - Everything in UNIX can be treated as a file
  - Linux supports multiple file systems
- Device control
  - Every system operation maps to a physical device
    - Few exceptions: CPU, memory, etc.
- Networking
  - Handles packets
  - Handles routing and network address resolution issues

Splitting the Kernel



Loadable Modules

- The ability to add and remove kernel features at runtime
- Each unit of extension is called a *module*
- Use **`insmod`** program to add a kernel module
- Use **`rmmod`** program to remove a kernel module

Classes of Devices and Modules

- Character devices
- Block devices
- Network devices
- Others

Character Devices

- Abstraction: a stream of bytes
  - Examples
    - Text console (**`/dev/console`**)
    - Serial ports (**`/dev/ttyS0`**)
  - Usually supports **`open`**, **`close`**, **`read`**, **`write`**
  - Accessed sequentially (in most cases)
  - Might not support file seeks
  - Exception: frame grabbers
    - Can access acquired image using **`mmap`** or **`lseek`**

Block Devices

- Abstraction: array of storage blocks
- However, applications can access a block device in bytes
  - Block and char devices differ only at the kernel level
  - A block device can host a file system

Network Devices

- Abstraction: data packets
- Send and receive packets
  - Do not know about individual connections
- Have unique names (e.g., **`eth0`**)
  - Not in the file system
  - Support protocols and streams related to packet transmission (i.e., no **`read`** and **`write`**)

Other Classes of Devices

- Examples that do not fit to previous categories:
  - USB
  - SCSI
  - FireWire
  - MTD

File System Modules

- Software drivers, not device drivers
- Serve as a layer between user API and block devices
- Intended to be device-independent

Security Issues

- Deliberate vs. incidental damage
- Kernel modules present possibilities for both
- System does only rudimentary checks at module load time
- Relies on limiting privilege to load modules
  - And trusts the driver writers
- Driver writer must be on guard for security problems

Security Issues

- Do not define security policies
  - Provide mechanisms to enforce policies
- Be aware of operations that affect global resources
  - Setting up an interrupt line
    - Could damage hardware
  - Setting up a default block size
    - Could affect other users

Security Issues

- Beware of bugs
  - Buffer overrun
    - Overwriting unrelated data
  - Treat input/parameters with utmost suspicion
  - Uninitialized memory
    - Kernel memory should be zeroed before being made available to a user
    - Otherwise, information leakage could result
      - Passwords

Security Issues

- Avoid running kernels compiled by an untrusted friend
  - Modified kernel could allow anyone to load a module

Version Numbering

- Every software package used in Linux has a release number
  - You need a particular version of one package to run a particular version of another package
  - Prepackaged distribution contains matching versions of various packages

Version Numbering

- Different throughout the years
- After version 1.0 but before 3.0
  - <major>.<minor>.<release>.<bugfix>
  - Time based releases (after two to three months)
- 3.x
  - Moved to 3.0 to commemorate 20th anniversary of Linux
  - <version>.<release>.<bugfix>
  - https://lkml.org/lkml/2011/5/29/204

License Terms

- GNU General Public License (GPL2)
  - GPL allows anybody to redistribute and sell a product covered by GPL
    - As long as the recipient has access to the source
    - And is able to exercise the same rights
  - Any software product derived from a product covered by the GPL be released under GPL

License Terms

- If you want your code to go into the mainline kernel
  - Must use a GPL-compatible license

Joining the Kernel Development Community

- The central gathering point
  - Linux-kernel mailing list
  - http://www.tux.org/lkml
- Chapter 20 of LKM further discusses the community and accepted coding style