I/O Management

- Stallings, Chapter 11
- 11.1 – 11.4, 11.7, 11.9 (and these notes!)
- Topics:
  - I/O devices
  - OS design issues
  - I/O buffering
  - Linux I/O features

Categories of I/O Devices

- Human readable
  - used to communicate with the user
  - video displays
  - keyboard
  - mouse
  - printer

Categories of I/O Devices

- Machine readable
  - used to communicate with electronic equipment
  - disk drives
  - flash drives
  - electronic controllers
  - robotic actuators

Categories of I/O Devices

- Communication
  - used to communicate with remote devices
  - secondary display devices
  - modems (mostly obsolete)
  - network interfaces

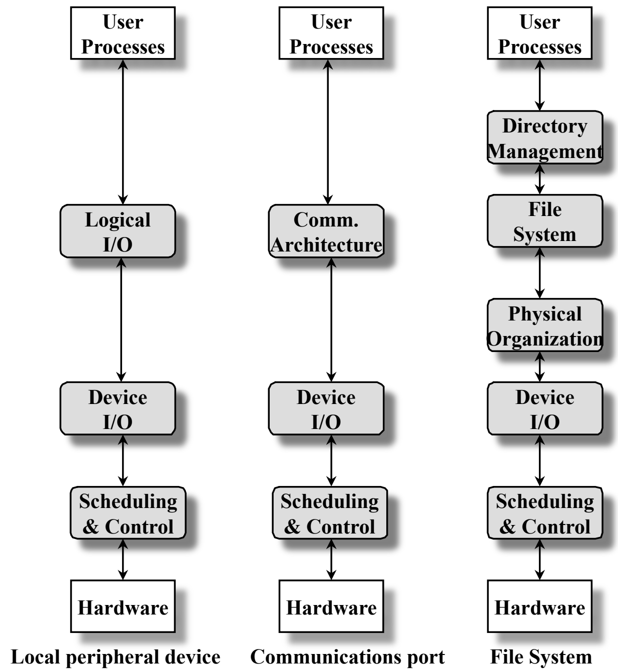
Operating System Design Issues

- Most I/O operations are extremely slow compared to main memory
- Use of multiprogramming allows for some processes to be waiting on I/O while another process executes
- Demand paging is used to bring in (pages of) additional "Ready" processes, but paging is also an I/O operation
- Hence, efficiency of I/O is an important issue

Operating System Design Issues HAL

- Generality is also an important issue
- Desirable to handle multiple (all?) I/O devices in a uniform manner
- Hide most of the details of device I/O in lower-level OS routines so that processes and upper levels see devices in general terms such as Read, Write, Open, and Close
- Leads to concept of "virtual" file system (VFS), which we will discuss with the file system

A Model of I/O Organization



(Kernel) I/O Buffering

- Reasons for buffering in kernel
  - Processes must wait for I/O to complete
  - Certain pages must remain in memory during I/O
- Block-oriented
  - information is stored in fixed sized blocks
  - transfers are made a block at a time
  - used for ssd, hard drive, cdrom, dvd (and formerly tapes)
- Character-oriented
  - transfer information as a stream of bytes
  - used for monitors, printers, network cards, communication ports, mouse, and many non-secondary-storage devices

Disk (Buffer) Cache

- Applies the concept of a cache memory to accessing disk and other block devices
  - Why especially important for block devices?
  - So, we buffer blocks of disk data in main memory
- When a disk block is requested, the OS first checks the buffer cache; if the block is present, it is returned, eliminating the need to access the disk.
- Although the buffer cache typically can grow and shrink, we occasionally need to replace blocks in the buffer with new blocks

LRU Replacement Strategy

- The block that has been in the cache the longest with no reference to it is replaced
- Logically, treat the cache as a list of pointers to blocks
  - At the front of the list is the most recently referenced block
  - When a block is referenced or brought into the cache, it is placed on the front of the list
- The block on the bottom of the stack is removed when a new block is brought in
- Wait! Isn't LRU to expensive to implement?

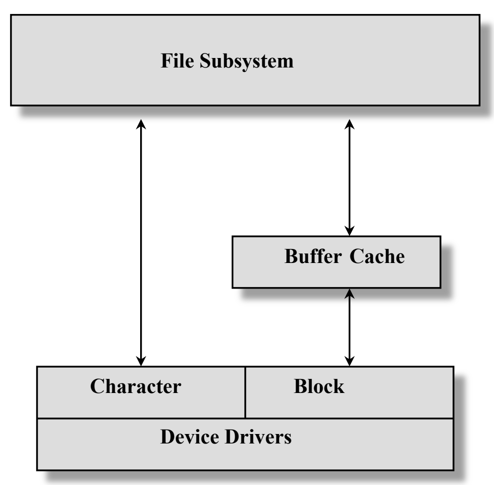
maintenance of LRU state is just a minor part of a system call

Least Frequently Used Replacement

- The block that has experienced the fewest references is replaced
- A counter is associated with each block
- Counter is incremented each time block is accessed
- Some blocks may be referenced many times in a short period of time and then not needed any more

Traditional Unix I/O Subsystem

- Files and I/O devices are both accessed through system calls to the file system.
- The buffer (disk) cache sits between the file system and (block) device drivers
- Character devices can be accessed directly through device files.



Unix/Linux Device Drivers

- Two types of devices: character vs. block
- Examples of each?

  Block - Hard drives, CDROM, tape SSD

  Character - everything else

- Usually, provide interfaces for at least the following system calls: open, close, read, write, (ioctl)
- Historically, device drivers in Unix were compiled statically into the kernel
- More recently (Linux), drivers are configured as dynamically loadable modules
- Advantage?

UNIX I/O Devices and Device Files

- Each I/O device had a device driver associated with it
- Each device also as a special device file associated with it
- Device drivers can be accessed via the device files, as with regular files
- Device files are one of the most elegant features of Unix
- All devices using the same driver have the same major device number
- Devices are distinguished by minor device number

Example Device Files (older Linux)

```
total 0
crw------- 1 root root    252,   0 Nov 23 21:46 hidraw0
crw------- 1 root root    252,   1 Nov 23 21:46 hidraw1
crw------- 1 root root     10, 228 Jul 16 06:32 hpet
crw------T 1 root root    108,   0 Jul 16 06:32 ppp
crw------- 1 root root     10,   1 Jul 16 06:32 psaux
brw-rw---T 1 root disk      8,   0 Jul 16 10:32 sda
brw-rw---T 1 root disk      8,   1 Jul 16 10:32 sda1
brw-rw---T 1 root disk      8,   2 Jul 16 10:32 sda2
brw-rw---T 1 root disk      8,   5 Jul 16 10:32 sda5
brw-rw---T 1 root disk      8,   6 Jul 16 10:32 sda6
brw-rw---T 1 root disk      8,   7 Jul 16 10:32 sda7
brw-rw---T 1 root disk      8,   8 Jul 16 10:32 sda8
brw-rw---T 1 root disk      8,   9 Jul 16 10:32 sda9
crw-rw-rw- 1 root root      5,   0 Nov 26 09:47 tty
crw------- 1 root root      4,   0 Jul 16 06:32 tty0
crw-rw---- 1 root tty       4,   1 Jul 16 10:32 tty1
crw-rw---- 1 root tty       4,   2 Jul 16 10:32 tty2
crw-rw---- 1 root tty       4,   3 Jul 16 10:32 tty3
crw-rw---- 1 root tty       4,   4 Jul 16 10:32 tty4
crw------- 1 root root      7,   0 Jul 16 06:32 vcs
crw------- 1 root root      7,   1 Jul 16 06:32 vcs1
crw------- 1 root root      7,   2 Jul 16 10:32 vcs2
crw------- 1 root root      7,   3 Jul 16 10:32 vcs3
crw------- 1 root root      7,   4 Jul 16 10:32 vcs4
```


Features of Linux Drivers

- Kernel code - even though drivers are often added to the system for new devices, by third parties, they are kernel code and, if buggy, can easily crash the system or worse.
- Kernel interfaces - must provide a standard interface to Linux kernel or subsystem (file I/O interface, SCSI interface, etc)
- Kernel mechanisms - make use of standard kernel services, such as wait queues
- Most drivers can be configured as modules, so they are demand loadable as well as boot configurable. If driver is present but hardware is not, no problem.
- Drivers may use DMA for data transfers between an adapter card and main memory

Summary

- Wide variety of peripheral devices
- Many devices accessed via (virtual) file system
- DMA very common communication mechanism
- Character and block devices handled differently
  - For block devices, data read through buffer cache
- Device numbers
  - Major number identifies driver
  - Minor number identifies specific (logical/physical) device
- Nowadays, in Linux most drivers are implemented as loadable modules