# Operating Systems

## Lecture #4: IO Management

Written by David Goodwin
based on the lecture series of Dr. Dayou Li
and the book *Understanding Operating Systems* $4^{th}$ *ed.*
by *I.M.Flynn and A.McIver McHoes* (2006)

Department of Computer Science and Technology,
University of Bedfordshire.

Operating Systems, 2013

$18^{th}$ February 2013

# Outline

# Device types

# I/O HARDWARE



- Incredible variety of I/O devices
- Common concepts
    - Port
    - Bus (daisy chain or shared direct access)
    - Controller (host adapter)
- I/O instructions control devices
- Devices have addresses, used by
    - Memory-mapped I/O
    - Direct I/O instructions

# TYPES OF DEVICES



David Goodwin
University of Bedfordshire



- Dedicated
  - Assigned to only one job at a time, e.g. printers, tape drives, etc.
- Shared
  - Can be assigned to several processes, e.g. a disk can be shared by several processes at the same time by interleaving their requests which are carefully controlled by the Device Manager.
- Virtual
  - A combination of the first two – dedicated devices transferred to shared devices, e.g. a printer made sharable through spooling (Simultaneous Peripheral Operation On-Line).

# STORAGE DEVICES

- Sequential access storage
  - Paper printout, punch cards, paper tapes, magnetic tapes
- Direct access storage devices (DASDs)
  - Magnetic disks (fixed-head and moveable-head)
  - Optical discs (CD-ROM, CD-Recordable, CD-Rewritable, DVD)
  - Magneto-optical (combination of magnetic and optical discs)
- Flash memory
  - Removable medium that emulates random access memory but unlike RAM, it stores data securely even it is removed from its power supply
- RAID (Redundant Array of Inexpensive/independent Disks)
  - A set of physical disk drives viewed as a single logical unit by OS to close the widening gap between increasingly fast processors and slower disk drives

# DASD

# DASD ACCESS TIMES

- For a fixed-head DASD
  Total access time$(t_a) =$
  Sum of search time$(t_s)$ + Transfer time$(t_t)$
  - Search time $t_s$
    Best search time $t_s = 0$
    Worst search time $t_s = r$ ($r$ is time taken per revolution of disk)
    Average search time $t_s = \frac{r}{2}$
  - Transfer time $t_t$

# DASD ACCESS TIMES

- Example for a fixed-head DASD

  A complete revolution takes 14ms, and data transfer time is 0.16ms per record. Calculate
  - Total average access time for 20 individual records
  - Total average access time for a block of 20 records
  - Time saving between the two

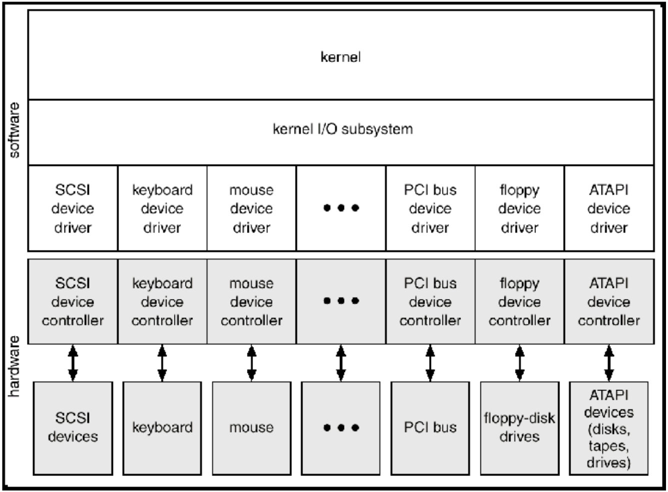
  - $t_a$ for 20 individual records $= 20\left(\frac{14}{2} + 0.16\right) = 143.2ms$
  - $t_a$ for block of 20 records $= \frac{14}{2} + 20 \times 0.16 = 10.2ms$
  - Time saving $= 143.2 – 10.2 = 133.0ms$

# Components of IO subsystem

# APPLICATION I/O INTERFACE

- I/O system calls encapsulated device behaviours in generic classes
- Device-driver layer hides differences among I/O controllers from kernel
- Devices vary in many dimensions
  - Character-stream or block
  - Sequential or random-access
  - Synchronous or asynchronous
  - Sharable or dedicated
  - Speed of operation
  - Read-write, read only, or write only
- I/O subsystem is independent of hardware, which simplifies job for OS developers; device manufacturers have to develop device drivers to suit a specific OS

# A KERNEL I/O STRUCTURE



# CHARACTERISTICS OF I/O DEVICES



| aspect | variation | example |
|---|---|---|
| data-transfer mode | character<br>block | terminal<br>disk |
| access method | sequential<br>random | modem<br>CD-ROM |
| transfer schedule | synchronous<br>asynchronous | tape<br>keyboard |
| sharing | dedicated<br>sharable | tape<br>keyboard |
| device speed | latency<br>seek time<br>transfer rate<br>delay between operations | |
| I/O direction | read only<br>write only<br>readÐwrite | CD-ROM<br>graphics controller<br>disk |

# BLOCK AND CHARACTER I/O DEVICES





- Block devices include disk drives
  - Commands include read, write, seek
  - Raw I/O or file-system access
  - Memory-mapped file access possible
- Character devices include keyboards, mice, serial ports
  - Commands include get, put
  - Libraries layered on top allow line editing

# BLOCKING AND NON-BLOCKING I/O

- Blocking - process suspended until I/O completed
  - Easy to use and understand
  - Insufficient for some needs
- Nonblocking - I/O call returns as much as available
  - User interface, data copy (buffered I/O)
  - Implemented via multi-threading
  - Returns quickly with count of bytes read or written
  - Processing and displaying data on screen while receiving keyboard/mouse input; video application reading frames from file on a disk while decompressing and displaying the output.
- Asynchronous - process runs while I/O executes
  - Difficult to use
  - I/O subsystem signals process when I/O completed

# KERNEL I/O SUBSYSTEM



- Scheduling
  - Some I/O request ordering via per-device queue
  - Some operating systems try fairness
- Buffering - store data in memory while transferring between devices
  - To cope with device speed mismatch
  - To cope with device transfer size mismatch
  - To maintain "copy semantics"

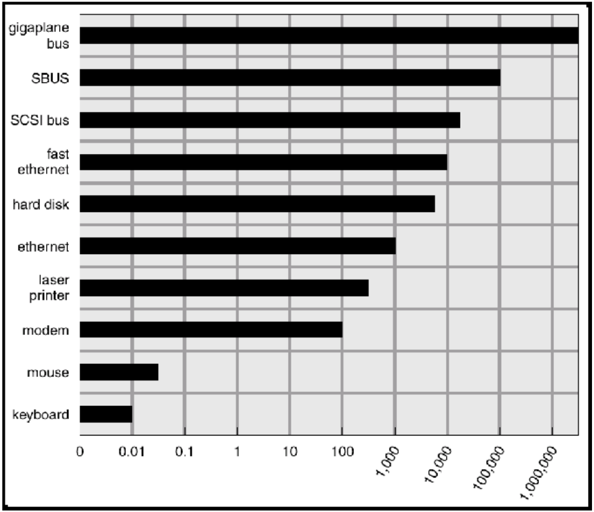
# KERNEL I/O SUBSYSTEM



- Caching - fast memory holding copy of data
  - Always just a copy
  - Key to performance
- Spooling (simultaneous peripheral operation on-line) - buffer that holds output for a device
  - If device can serve only one request at a time
  - i.e. printing
- Device reservation - provides exclusive access to a device
  - System calls for allocation and de-allocation
  - Watch out for deadlock

# SUN ENTERPRISE 6000 DEVICE TRANSFER RATES





# Components of the IO system

- **IO subsystem**'s components perform similar functions
- The channel's job is to keep up with the IO requests from the CPU and pass them to the control unit.
- **IO channels** are programmable units places btween the CPU and the control units
  - their job is to synchronize the fast speed of the CPU with the slow speed of the IO device
  - make it possible to overlap IO operations with processor operations so the IO and CPU can process concurrently
- Channels use **IO channel programs**
  - Each channel program specifies the action to be performed by the devices
  - controls the transmission of data between main memory and the control units
- **IO control unit** interprets the signal sent from the channel
  - control unit is sometimes part of the device
  - one control unit may be attached to several similar devices

# Components of the IO system

- At the start of an IO command, the information passed from the CPU to the channel is:
  - IO command (READ, WRITE, REWIND, etc.)
  - Channel number
  - Address of the physical record to be transferred (from secondary storage)
  - Starting address of a memory buffer from which or into which the record is to be transferred
- Because the channels are as fast as the CPU, they can direct several control units by interleaving commands
- Each control unit can direct several devices in a similar way
- Channels are often shared because they are the most valuable items in terms of the IO subsystem

# Communication

# Communication among devices





- The Device Manager relies on several auxiliary features to keep running efficiently
- There are three problems that must be solved:
  1. It needs to know which components are busy and which are free
  2. It must be able to accommodate the requests that come in during heavy IO traffic
  3. It must accommodate the disparity of speeds between the CPU and the IO devices
- The first is solved by structuring the interaction between units.
- The last two are solved by buffering records and queueing requests
- The success of the operation depends on the system's ability to know when a device has completed an operation

# Communication among devices



- This is done by hardware flagging, and must be tested by the CPU
  - The flag is made up of three bits and resides in the **Channel Status Word**, or CSW.
  - It is in a predefined location in main memory and contains information indicating the status of the channel.
  - each bit represents one of the components of the IO subsystem
    - one for the channel
    - one for the control unit
    - one for the device
  - Each bit is changed from 0 to 1 to indicate that the unit has changed from free to busy
  - Each component has access to the flag
  - The flag is tested before proceeding with the next IO operation to ensure that the entire path is free
  - There are two common ways to perform this test: **polling** and **interrupts**.

# POLLING





- Determines state of device
  - command-ready
  - busy
  - Error
- Example:
  - CPU periodically tests the channel status bit (in CSW)
  - If the channel is busy, the CPU performs some other task
  - Else, the channel performs the IO operation
- Busy-wait cycle to wait for I/O from device
- Inefficient when attempted repeatedly, yet rarely finds a device ready for service while other CPU processing remains undone; More efficient for the hardware controller to notify the CPU when it is ready for service rather than to require the CPU to poll repeatedly for an I/O completion.

# INTERRUPTS

- More efficient way to test the flag than polling
  - A hardware mechanism does the test as part of every machine instruction executed by the CPU
  - If the channel is busy the flag is set so that execution of the instructionsis automatically interupted
  - control is passed to the interrupt handler
    - part of the OS which resides in a predefined location of the memory
    - find out which unit sent the signal
    - analyse it's status and restart is where appropriate with the next operation
    - return control to the interrupted process
  - Some systems have hardware that can distinguish between several types of interrupts
    - these interrupts are ordered by priority
    - each one can transfer control to a corresponding memory location
    - hardware will automatically intercept all interrupts of similar or lower priority

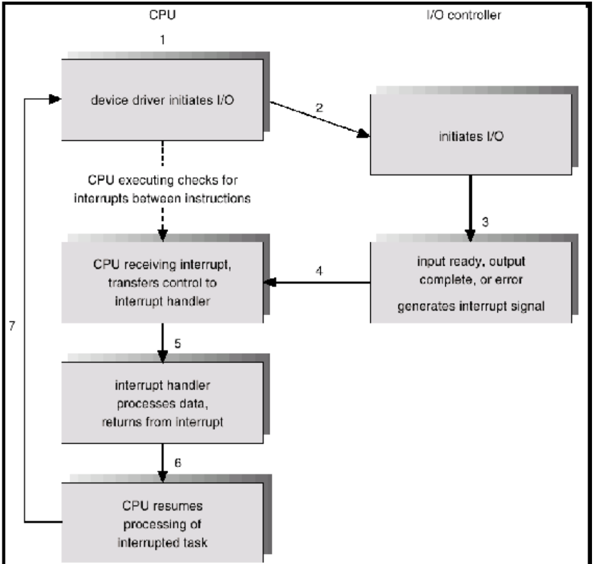
# INTERRUPTS



- CPU Interrupt-request line triggered by I/O device
- Interrupt handler receives interrupts
- Maskable to ignore or delay some interrupts
- Interrupt vector to dispatch interrupt to correct handler
  - Based on priority
  - Some unmaskable
- Interrupt mechanism also used for exceptions

# INTERRUPT-DRIVEN I/O CYCLE





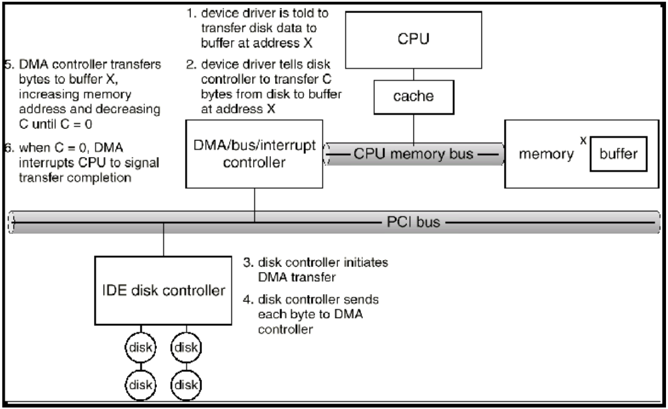
# DIRECT MEMORY ACCESS

- Avoid programmed I/O (PIO) for large data movement
- Bypasses CPU to transfer data directly between I/O device and memory
  - DMA controller and CPU can compete for the system bus if they need it at the same time
    - Host writes a DMA command block in memory
    - Command block consisting of a pointer to the source of transfer, a pointer to the destination (memory address), and a count of the number of bytes to be transferred
    - CPU can then go to another task while the control unit completes the transfer independently
    - DMA controller then sends and interrupt to the CPU when the operation is complete
  - Without the DMA the CPU is responsible for this transfer of data - which is time-consuming and creates significant overheads, reducing the performance of the CPU
- Some computer systems use direct virtual-memory access (DVMA)

# SIX-STEP PROCESS TO PERFORM DMA TRANSFER







# Buffers

- Used extensively to better synchronise the movement of data between slow io devices and fast CPU
- Temporary storage areas residing in three locations
  - Main memory
  - Channels
  - Control units
- Used to store data read from an input device before it's needed by the processor or that will be written to an output device
- To decrease idle time for devices and maximise their throughput, the technique of double buffering is used
  - two buffers are present in main memory, channels, and control units
  - record will be ready at any time, avoiding delay caused by waiting for a buffer to fill up with data
  - While one record is being processed by the CPU, another can be read or written by the channel

# Management of IO requests

# ERROR HANDLING



- OS can recover from disk read, device unavailable, transient write failures
- Most return an error number or code when I/O request fails
- System error logs hold problem reports

# KERNEL DATA STRUCTURES



David Goodwin
University of Bedfordshire



- Kernel keeps state information for I/O components, including open file tables, network connections, character device state
- Many, many complex data structures to track buffers, memory allocation, “dirty” blocks
- Some use object-oriented methods and message passing to implement I/O

# I/O REQUEST TO HARDWARE OPERATIONS

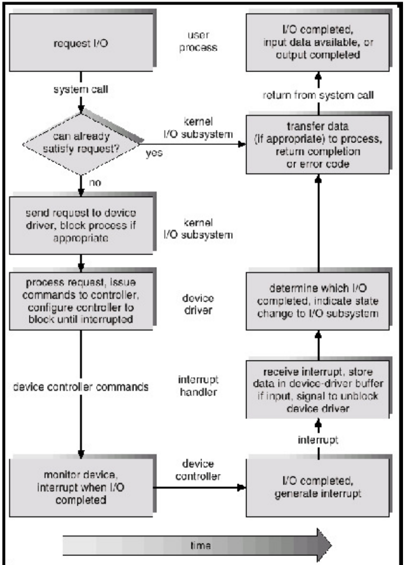


- Consider reading a file from disk for a process:
  - Determine device holding the file
  - Translate name to device representation
  - Physically read data from disk into buffer
  - Make data available to requesting process
  - Return control to process

# LIFE CYCLE OF AN I/O REQUEST







# PERFORMANCE

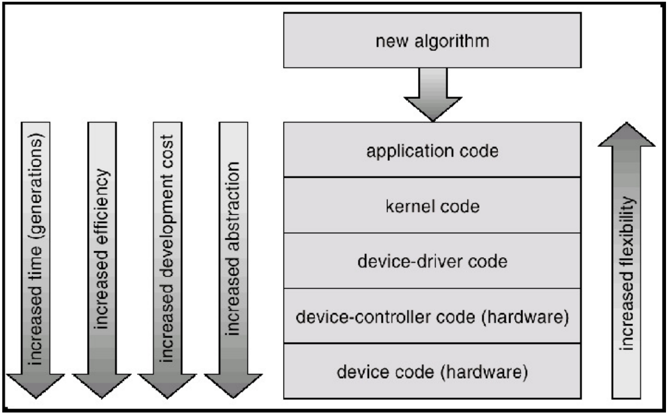
- I/O is a major factor in system performance
  - Demands CPU to execute device driver, kernel I/O code
  - Context switches due to interrupts
  - Data copying
  - Network traffic especially stressful

# IMPROVING PERFORMANCE

- Reduce number of context switches
- Reduce data copying
- Reduce interrupts by using large transfers, smart controllers, polling
- Use DMA
- Balance CPU, memory, bus, and I/O performance for highest throughput

# DEVICE-FUNCTIONALITY PROGRESSION





# IO Traffic Controller

- The **IO traffic controller** monitors the status of every device, control unit, and channel.
- Three main tasks:
  1. determine if there's at least one path available
  2. if more than one path, determines which to select
  3. if paths are all busy, determines when one will become available
- Maintains a database containing status and connections for each unit in the IO subsystem grouped as:
  1. Channel Control Blocks
  2. Control Unit Control Blocks
  3. Device Control Blocks
- To choose a free path; traces backward from the control block of the requested device through the control units to the channels
- If no path available the Process is linked to the queues kept in the Control Blocks
- Creates multiple wait queues with one queue per path

# IO Scheduler & IO Device Handler



- **IO scheduler** allocates devices, control units, and channels and must decide which which requests to satisfy first.
- **IO device handler** processes the IO interrupts, handles error conditions, and provides detailed scheduling algorithms

# Device Handling Algorithms

# Seek strategy

- Allocate access to the device among the many processes that may be waiting for it
- Determines the order in which the processes get the device
- Goal to keep seek time to a minimum
- Some common strategies include:
  - first-come, first-served (FCFS)
  - shortest seek time first (SSTF)
  - SCAN (with varients; LOOK, N-Step SCAN, C-SCAN, C-LOOK)
- Every scheduling algorithm should do the following:
  1. Minimise arm movement
  2. Minimise mean response time
  3. Minimise the variance in response time

# First-come, First-served

- Simplest device-scheduling algorithm; easy to program and essentially fair to users
- However, on average, it doesn't meet any of the three goals of a seek strategy
- FCFS has the disadvantage of extreme arm movement, and hence large seek time

# Shortest Seek Time First

- With SSTF, the request with the track closest to the one being served is the next to be satisfied
- Has the disadvantage that it favours easy-to-reach requests and postpones travelling to those that are a long way out of the way
- meets the first goal of seek strategies, but fails to meet the other two

# SCAN

- SCAN uses a directional bit to indicate whether the arm is moving toward the centre of the disk or away from it
- Moves the arm from outer to inner track, servicing every request in its path, then reverses direction
- There are a number of variations of the SCAN algorithm:
  - LOOK
    - doesn't go all the way to the edge if there are no requests there
  - N-Step SCAN
    - doesn't incorporate requests into the arm's path as it travels, but holds all requests until the arm starts on its way back
  - C-SCAN (circular SCAN)
    - instead of reversing direction when the arm meets the innermost track, the arm immediately returns to the outermost track
  - C-LOOK (circular LOOK)
    - an optimisation of C-SCAN, just as LOOK is and optimisation of SCAN

# Comparison of seek strategies



**FCFS** works well with light loads, but as soon as the load grows, service time becomes unacceptably long.

**SSTF** works well with moderate loads but has the problem of localisation under heavy loads.

**SCAN** works well with light to moderate loads and eliminates the problem of indefinite postponement.

**C-SCAN** works well with moderate to heavy loads and has a very small variance in service times

# RAID

# RAID

- RAID = Redundant Array of Independent Disks
  - a group of hard disks controlled in such a way that they speed read access of data on secondary storage devices and aid data recovery
  - viewed as a single operating unit by OS
  - RAID assumes that several smaller-capacity disk drives are preferable to a few large capacity disk drives
  - system can simultaneously access and request data from multiple drives
  - improves IO performance and data recovery in the event of disk failure
- Data is divided into segments called **strips**
  - Strips are distributed accross the disks in the array
  - A set of consecutive strips is called a **stripe**
  - The whole process is called striping

# Levels of RAID

| RAID level | Error Correction | IO request rate | Data transfer rate |
|---|---|---|---|
| 0 | none | excellent | excellent |
| 1 | mirroring | R:good, W:fair | R:fair W:fair |
| 2 | Hamming code | poor | excellent |
| 3 | Word parity | poor | excellent |
| 4 | Strip parity | R:excellent, W:fair | R:fair, W:poor |
| 5 | Distributed strip parity | R:excellent, W:fair | R:fair, W:poor |
| 6 | Distributed strip parity and independent data checking | R:excellent, W:poor | R:fair, W:poor |

# SUMMARY

# Key Terms

access time
blocking
buffers
channel status word (CSW)
C-LOOK
C-SCAN
direct access storage Device (DASD)
direct memory access (DMA)
first-come, first-served (FCFS)
IO channel
IO channel program
IO contol unit
IO device handler
IO scheduler

# Key Terms

IO subsystem
IO traffic controller
interrupt
LOOK
N-step SCAN
parity bit
RAID
SCAN
search
search time
seek strategy
seek time
shortest seek time first (SSTF)