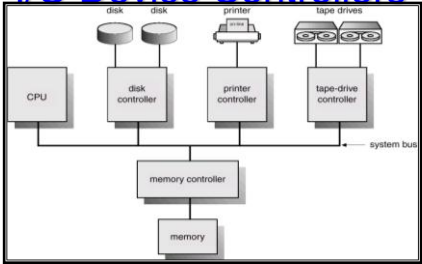


I/O Systems

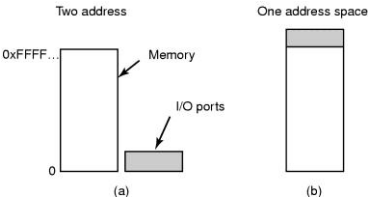
CS 256/456  
Dept. of Computer Science, University  
of Rochester

I/O Device Controllers



- I/O devices have both mechanical component & electronic component
- The electronic component is the device controller
  - It contains control logic, command registers, status registers, and on-board buffer space

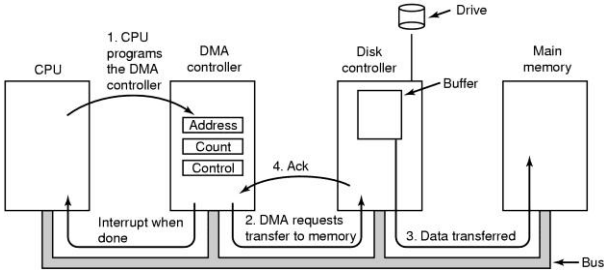
I/O Ports & Memory-Mapped I/O



- I/O methods:
- Separate I/O and memory space; special I/O commands (IN/OUT)
  - Memory-mapped I/O

- Issues with them:
- Convenience/efficiency when using high-level language;
  - Protection mechanisms;
  - Special data access schemes: TEST
  - Data caching

Direct Memory Access (DMA)



- Are the addresses CPU sends to the DMA controller virtual or physical addresses?
- Can the disk controller directly read data into the main memory (bypassing the controller buffer)?

## How is I/O accomplished?

- Polling-based
  - CPU spins and polls the I/O until it completes
- Periodic polling
  - Continuous polling consumes too much CPU
  - Instead, we poll periodically - saving CPU overhead, may not react immediately to hardware events
- Interrupt-driven
  - CPU initiates I/O and then does something else; gets notified when the I/O is done (interrupts)

10/20/2010

CSC 2/456

5

## Interrupt Handlers

1. Save registers of the old process
2. Set up context for interrupt service procedure (switch from the user space to kernel space: MMU, stack, ...)
3. Run service procedure; when safe, re-enable interrupts
4. Run scheduler to choose the new process to run next
5. Set up context (MMU, registers) for process to run next
6. Start running the new process

How much cost is it? Is it a big deal?

For Gigabit Ethernet, each packet arrives once every 12us.

10/20/2010

CSC 2/456

6

## Interrupt Vectors

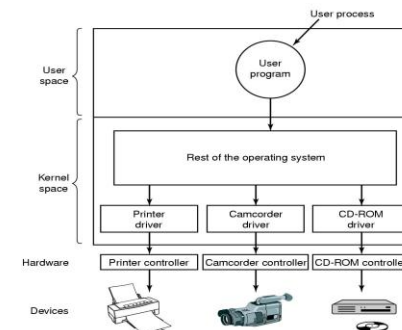
- Intel Pentium processor event-vector table
  - 0: divide by zero
  - 6: invalid opcode
  - 11: segment not present
  - 12: stack fault
  - 14: page fault
  - ...31: non-maskable
  - 32-255: maskable interrupts

10/20/2010

CSC 2/456

7

## I/O Software Layers



- Device-dependent OS I/O software; directly interacts with controller hardware
- Interface to upper-layer OS code is standardized

10/20/2010

CSC 2/456

8

### Device Driver Reliability

- Device driver is the device-specific part of the kernel-space I/O software; It also includes interrupt handlers
- Device drivers must run in kernel mode
  - ⇒ The crash of a device driver typically brings down the whole system
- Device drivers are probably the buggiest part of the OS
- How to make the system more reliable by isolating the faults of device drivers?
  - Run most of the device driver code at user level
  - Restrict and limit device driver operations in the kernel

10/20/2010

CSC 2/456

9

### High-level I/O Software

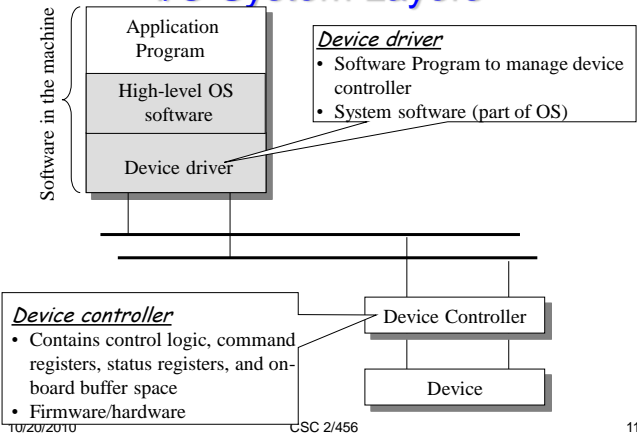
- Device independence
  - reuse software as much as possible across different types of devices
- Buffering
  - data coming off a device is stored in intermediate buffer
  - access speed/granularity matching with I/O devices
- caching
- speculative I/O

10/20/2010

CSC 2/456

10

### I/O System Layers

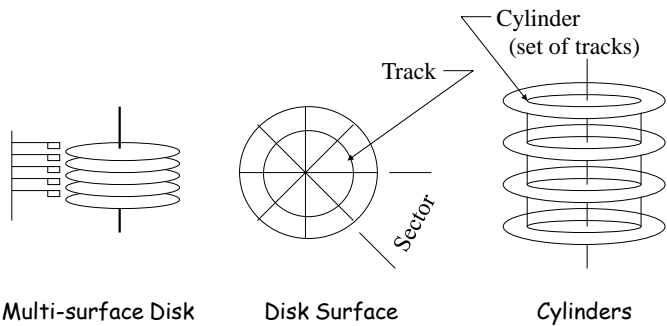


10/20/2010

CSC 2/456

11

### Disk Drive – Mechanical Parts



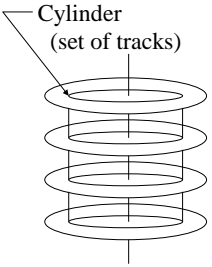
10/20/2010

CSC 2/456

12

### Disk Structure

- Disk drives are addressed as large 1-dimensional arrays of *logical blocks*, where the logical block is the smallest unit of transfer.
- The 1-dimensional array of logical blocks is mapped into the sectors of the disk sequentially.
  - Sector 0 is the first sector of the first track on the outermost cylinder.
  - Mapping proceeds in order through that track, then the rest of the tracks in that cylinder, and then through the rest of the cylinders from outermost to innermost.



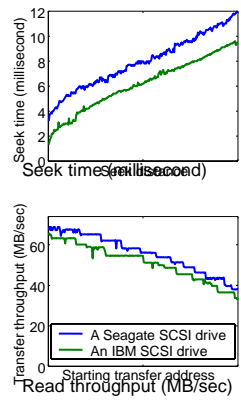
Cylinder (set of tracks)

10/20/2010

13

### Disk Performance Characteristics

- A disk operation has three major components
  - Seek* - moving the heads to the cylinder containing the desired sector
  - Rotation* - rotating the desired sector to the disk head
  - Transfer* - sequentially moving data to or from disk



Seek time (millisecond)

Transfer throughput (MB/sec)

Read throughput (MB/sec)

Starting transfer address

— A Seagate SCSI drive  
— An IBM SCSI drive

10/20/2010

CSC 2/456

14

### Disk Scheduling

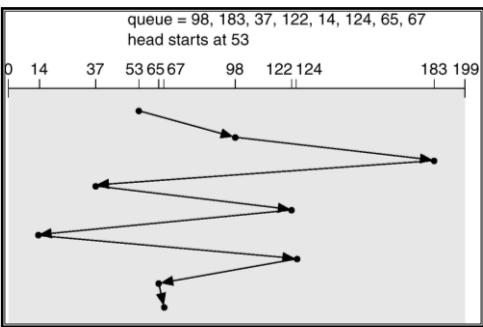
- Disk scheduling - choose from outstanding disk requests when the disk is ready for a new request
  - can be done in both disk controller and the operating system
  - Disk scheduling non-preemptible
- Goals of disk scheduling
  - overall efficiency - small resource consumption for completing disk I/O workload
  - fairness - prevent starvation

10/20/2010

CSC 2/456

15

### FCFS (First-Come-First-Serve)



queue = 98, 183, 37, 122, 14, 124, 65, 67  
head starts at 53

0 14 37 53 65 67 98 122 124 183 199

- Illustration shows the total head movement is 640.
- Starvation?

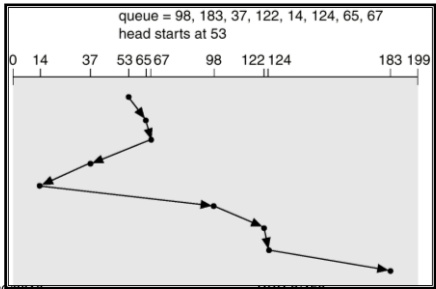
10/20/2010

CSC 2/456

16

### SSTF (Shortest-Seek-Time-First)

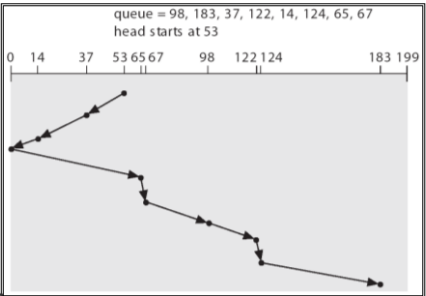
- Selects the request with the minimum seek time from the current head position.
- SSTF scheduling is a form of SJF scheduling.
- Illustration shows the total head movement is 236.



Starvation?

### SCAN

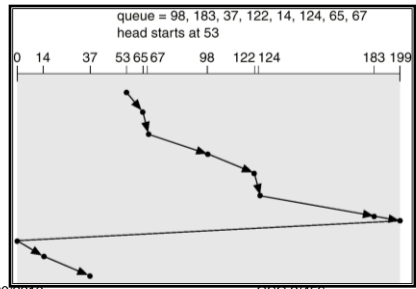
- The disk arm starts at one end of the disk, and moves toward the other end, servicing requests until it gets to the other end, where the head movement is reversed and servicing continues.
- Sometimes called the *elevator algorithm*.
- Illustration shows the total head movement is 208.



Starvation?

### C-SCAN (Circular-SCAN)

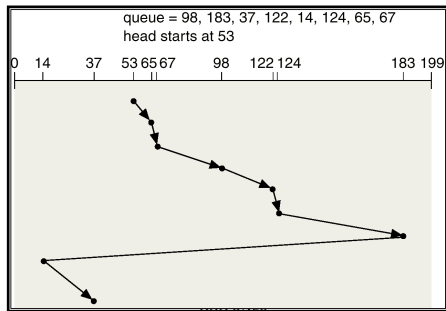
- Provides a more uniform wait time than SCAN.
- The head moves from one end of the disk to the other, servicing requests as it goes. When it reaches the other end, however, it immediately returns to the beginning of the disk, without servicing any requests on the return trip.



Starvation?

### C-LOOK

- Variation of C-SCAN
- Arm only goes as far as the last request in each direction, then reverses direction immediately, without first going all the way to the end of the disk.

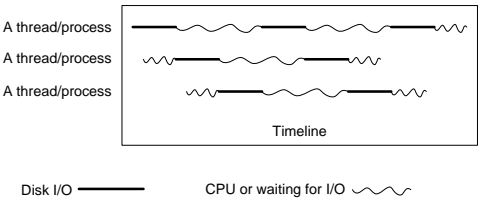


### Deadline Scheduling in Linux

- A regular elevator-style scheduler similar to C-LOOK
- Additionally, all I/O requests are put into a FIFO queue with an expiration time (e.g., 500ms)
- When the head request in the FIFO queue expires, it will be executed next (even if it is not next in line according to C-LOOK).
- A mix of performance and fairness.

### Concurrent I/O

- Consider two request handlers in a Web server
  - each accesses a different stream of sequential data (a file) on disk;
  - each reads a chunk (the buffer size) at a time; does a little CPU processing; and reads the next chunk
- What happens?



### How to Deal with It?

- Aggressive prefetching
- Anticipatory scheduling [Iyer & Druschel, SOSP 2001]
  - at the completion of an I/O request, the disk scheduler will wait a bit (despite the fact that there is other work to do), in anticipation that a new request with strong locality will be issued; schedule another request if no such new request appears before timeout
  - included in Linux 2.6

### Exploiting Concurrency

- RAID: Redundant Arrays of Independent Disks
  - RAID 0: data striping at block level, no redundancy
  - RAID 1: mirrored disks (100% overhead)
  - RAID 2: bit-level striping with parity bits, synchronized writes
  - RAID 3: data striping at the bit level with parity disk, synchronized writes
  - RAID 4: data striping at block level with parity disk
  - RAID 5: scattered parity
  - RAID 6: handles multiple disk failures

## Disk Management

- Formatting
  - Header: sector number etc.
  - Footer/tail: ECC codes
  - Gap
  - Initialize mapping from logical block number to defect-free sectors
- Logical disk partitioning
  - One or more groups of cylinders
  - Sector 0: master boot record loaded by BIOS firmware, which contains partition information
  - Boot record points to boot partition

10/20/2010

CSC 2/456

25

## Swap Space Management

- Part of file system?
  - Requires navigating directory structure
  - Disk allocation data structures
- Separate disk partition
  - No file system or directory structure
  - Optimize for speed rather than storage efficiency
  - When is swap space created?

10/20/2010

CSC 2/456

26

## Disclaimer

- Parts of the lecture slides contain original work of Abraham Silberschatz, Peter B. Galvin, Greg Gagne, Andrew S. Tanenbaum, and Gary Nutt. The slides are intended for the sole purpose of instruction of operating systems at the University of Rochester. All copyrighted materials belong to their original owner(s).

10/20/2010

CSC 2/456

27