I/O Systems
I/O Systems

- I/O Hardware
- Application I/O Interface
- Kernel I/O Subsystem
- Transforming I/O Requests to Hardware Operations
- Streams
- Performance
Objectives

- Explore the structure of an operating system’s I/O subsystem
- Discuss the principles of I/O hardware and its complexity
- Provide details of the performance aspects of I/O hardware and software
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
  - Direct I/O instructions
  - Memory-mapped I/O
A Typical PC Bus Structure

- Monitor
- Processor
- Graphics controller
- Bridge/mem controller
- Cache
- Memory
- IDE disk controller
- Disk
- Expansion bus interface
- Expansion bus
- Parallel port
- Serial port
- SCSI controller
- Disk
- Keyboard
### Device I/O Port Locations on PCs (partial)

<table>
<thead>
<tr>
<th>I/O address range (hexadecimal)</th>
<th>device</th>
</tr>
</thead>
<tbody>
<tr>
<td>000–00F</td>
<td>DMA controller</td>
</tr>
<tr>
<td>020–021</td>
<td>interrupt controller</td>
</tr>
<tr>
<td>040–043</td>
<td>timer</td>
</tr>
<tr>
<td>200–20F</td>
<td>game controller</td>
</tr>
<tr>
<td>2F8–2FF</td>
<td>serial port (secondary)</td>
</tr>
<tr>
<td>320–32F</td>
<td>hard-disk controller</td>
</tr>
<tr>
<td>378–37F</td>
<td>parallel port</td>
</tr>
<tr>
<td>3D0–3DF</td>
<td>graphics controller</td>
</tr>
<tr>
<td>3F0–3F7</td>
<td>diskette-drive controller</td>
</tr>
<tr>
<td>3F8–3FF</td>
<td>serial port (primary)</td>
</tr>
</tbody>
</table>
Polling

- Determines state of device
  - command-ready
  - busy
  - Error

- **Busy-wait** cycle to wait for I/O from device
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 **nonmaskable**
- Interrupt mechanism also used for exceptions
Interrupt-Driven I/O Cycle

1. CPU
2. I/O controller initiates I/O
3. input ready, output complete, or error generates interrupt signal
4. CPU receiving interrupt, transfers control to interrupt handler
5. interrupt handler processes data, returns from interrupt
6. CPU resumes processing of interrupted task
7. CPU executing checks for interrupts between instructions
Direct Memory Access

- Used to avoid *programmed I/O* for large data movement
- Requires **DMA** controller
- Bypasses CPU to transfer data directly between I/O device and memory
Six Step Process to Perform DMA Transfer

1. Device driver is told to transfer disk data to buffer at address X

2. Device driver tells disk controller to transfer C bytes from disk to buffer at address X

3. Disk controller initiates DMA transfer

4. Disk controller sends each byte to DMA controller

5. DMA controller transfers bytes to buffer X, increasing memory address and decreasing C until C = 0

6. When C = 0, DMA interrupts CPU to signal transfer completion
Application I/O Interface

- I/O system calls encapsulate device behaviors 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
  - Sharable or dedicated
  - Speed of operation
  - read-write, read only, or write only
# A Kernel I/O Structure

<table>
<thead>
<tr>
<th>Hardware</th>
<th>Software</th>
</tr>
</thead>
<tbody>
<tr>
<td>SCSI devices</td>
<td>SCSI device controller</td>
</tr>
<tr>
<td>keyboard</td>
<td>keyboard device controller</td>
</tr>
<tr>
<td>mouse</td>
<td>mouse device controller</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
</tr>
<tr>
<td>PCI bus</td>
<td>PCI bus device controller</td>
</tr>
<tr>
<td>floppy-disk drives</td>
<td>floppy device controller</td>
</tr>
<tr>
<td>ATAPI devices (disks, tapes, drives)</td>
<td>ATAPI device controller</td>
</tr>
</tbody>
</table>

## Kernel I/O Subsystem

- kernel
  - SCSI device driver
  - keyboard device driver
  - mouse device driver
  - ... (ellipses)
  - PCI bus device driver
  - floppy device driver
  - ATAPI device driver
## Characteristics of I/O Devices

<table>
<thead>
<tr>
<th>aspect</th>
<th>variation</th>
<th>example</th>
</tr>
</thead>
<tbody>
<tr>
<td>data-transfer mode</td>
<td>character block</td>
<td>terminal disk</td>
</tr>
<tr>
<td>access method</td>
<td>sequential random</td>
<td>modem CD-ROM</td>
</tr>
<tr>
<td>transfer schedule</td>
<td>synchronous asynchronous</td>
<td>tape keyboard</td>
</tr>
<tr>
<td>sharing</td>
<td>dedicated sharable</td>
<td>tape keyboard</td>
</tr>
<tr>
<td>device speed</td>
<td>latency</td>
<td></td>
</tr>
<tr>
<td></td>
<td>seek time</td>
<td></td>
</tr>
<tr>
<td></td>
<td>transfer rate</td>
<td></td>
</tr>
<tr>
<td></td>
<td>delay between operations</td>
<td></td>
</tr>
<tr>
<td>I/O direction</td>
<td>read only</td>
<td>CD-ROM graphics controller</td>
</tr>
<tr>
<td></td>
<td>write only</td>
<td>disk</td>
</tr>
<tr>
<td></td>
<td>read–write</td>
<td></td>
</tr>
</tbody>
</table>
Block and Character 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
Network Devices

- Differ sufficiently from block and character to have own interface
- Unix and Windows NT/9x/2000 include socket interface
  - Separates network protocol from network operation
  - Includes `select` functionality
- Approaches vary widely (pipes, FIFOs, streams, queues, mailboxes)
Clocks and Timers

- Provide current time, elapsed time, timer
- **Programmable interval timer** used for timings, periodic interrupts
- `ioctl` (on UNIX) covers odd aspects of I/O such as clocks and timers
Blocking and Nonblocking 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

- **Asynchronous** - process runs while I/O executes
  - Difficult to use
  - I/O subsystem signals process when I/O completed
Two I/O Methods

(a) Synchronous

(b) Asynchronous
Kernel I/O Subsystem

- **Scheduling**
  - Some I/O request ordering via per-device queue
  - Some OSs attempt to provide 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”
## Device-status Table

<table>
<thead>
<tr>
<th>Device</th>
<th>Status</th>
<th>Request Details</th>
</tr>
</thead>
<tbody>
<tr>
<td>Keyboard</td>
<td>Idle</td>
<td></td>
</tr>
<tr>
<td>Laser Printer</td>
<td>Busy</td>
<td>Request for Laser Printer Address: 38546, Length: 1372</td>
</tr>
<tr>
<td>Mouse</td>
<td>Idle</td>
<td></td>
</tr>
<tr>
<td>Disk Unit 1</td>
<td>Idle</td>
<td></td>
</tr>
<tr>
<td>Disk Unit 2</td>
<td>Busy</td>
<td>Request for Disk Unit 2, File: xxx, Operation: Read, Address: 43046, Length: 20000</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

- File: yyy, Operation: Write, Address: 03458, Length: 500
Kernel I/O Subsystem

- **Caching** - fast memory holding copy of data
  - Always just a copy
  - Key to performance

- **Spooling** - hold 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 deallocation
  - Watch out for deadlock
Error Handling

- OS can recover from errors, such as disk read failure, device unavailable, transient write failures

- Most return an error number or code when I/O request fails

- System error logs hold problem reports
I/O Protection

- User process may accidentally or purposefully attempt to disrupt normal operation via illegal I/O instructions
  - All I/O instructions defined to be privileged
  - I/O must be performed via system calls
    - Memory-mapped and I/O port memory locations must be protected too
Use of a System Call to Perform I/O

1. trap to monitor
2. perform I/O
3. return to user

kernel

system call $n$

user program

case $n$
Kernel Data Structures

- Kernel keeps state info 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
UNIX I/O Kernel Structure

- **File System Record**
  - Inode Pointer
  - Pointer to Read and Write Functions
  - Pointer to Select Function
  - Pointer to IOCTL Function
  - Pointer to Close Function

- **Networking (Socket) Record**
  - Pointer to Network Info
  - Pointer to Read and Write Functions
  - Pointer to Select Function
  - Pointer to IOCTL Function
  - Pointer to Close Function

- **System-Wide Open-File Table**
  - Active-_inode Table
  - Network-Information Table

- **Per-Process Open-File Table**

- **User-Process Memory**

- **File Descriptor**
I/O Requests → Hardware Operations

- Consider reading a file from disk for a process:
  - Determine device holding 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

1. Request I/O
   - System call
   - Can already satisfy request?
     - Yes: Send request to device driver, block process if appropriate
     - No: Process request, issue commands to controller, configure controller to block until interrupted

2. User process
   - I/O completed, input data available, or output completed
   - Return from system call

3. Kernel I/O subsystem
   - Transfer data (if appropriate) to process, return completion or error code
   - Determine which I/O completed, indicate state change to I/O subsystem
   - Receive interrupt, store data in device-driver buffer if input, signal to unblock device driver

4. Device driver
   - I/O completed, generate interrupt

5. Monitor device, interrupt when I/O completed

Time
Performance

- I/O 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

- Increased time (generations)
- Increased efficiency
- Increased development cost
- Increased abstraction

Increased flexibility

- New algorithm
- Application code
- Kernel code
- Device-driver code
- Device-controller code (hardware)
- Device code (hardware)
Mass-Storage Structure
Chapter 10: Mass-Storage Structure

- Overview of Mass Storage Structure
- Disk Structure
- Disk Attachment
- Disk Scheduling
- Disk Management
- Swap-Space Management
- RAID Structure
- Disk Attachment
- Stable-Storage Implementation
- Tertiary Storage Devices
- Operating System Issues
- Performance Issues
Objectives

- Describe the physical structure of secondary and tertiary storage devices and the resulting effects on the uses of the devices
- Explain the performance characteristics of mass-storage devices
- Discuss operating-system services provided for mass storage, including RAID and HSM
Overview of Mass Storage Structure

- Magnetic disks provide bulk of secondary storage of modern computers
  - Drives rotate at 60 to 200 times per second
  - **Transfer rate** is rate at which data flows between drive and computer
  - **Positioning time** (random-access time) is time to move disk arm to desired cylinder (seek time) and time for desired sector to rotate under the disk head (rotational latency)
  - **Head crash** results from disk head making contact with the disk surface
    - That’s bad
- Disks can be removable
- Drive attached to computer via **I/O bus**
  - Busses vary, including **EIDE, ATA, SATA, USB, Fibre Channel, SCSI**
  - **Host controller** in computer uses bus to talk to **disk controller** built into drive or storage array
Moving-head Disk Mechanism

- Track $t$
- Spindle
- Sector $s$
- Cylinder $c$
- Platter
- Read-write head
- Arm assembly
- Arm
- Rotation
Overview of Mass Storage Structure (Cont.)

- **Magnetic tape**
  - Was early secondary-storage medium
  - Relatively permanent and holds large quantities of data
  - Access time slow
  - Random access ~1000 times slower than disk
  - Mainly used for backup, storage of infrequently-used data, transfer medium between systems
  - Kept in spool and wound or rewound past read-write head
  - Once data under head, transfer rates comparable to disk
  - 20-200GB typical storage
  - Common technologies are 4mm, 8mm, 19mm, LTO-2 and SDLT
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.
Disk Attachment

• Host-attached storage accessed through I/O ports talking to I/O busses

• SCSI itself is a bus, up to 16 devices on one cable, **SCSI initiator** requests operation and **SCSI targets** perform tasks
  ◦ Each target can have up to 8 **logical units** (disks attached to device controller)

• FC is high-speed serial architecture
  ◦ Can be switched fabric with 24-bit address space – the basis of **storage area networks (SANs)** in which many hosts attach to many storage units
  ◦ Can be **arbitrated loop (FC-AL)** of 126 devices
Network-Attached Storage

- Network-attached storage (NAS) is storage made available over a network rather than over a local connection (such as a bus).

\[
\text{NAS} \rightarrow \text{LAN/WAN} \rightarrow \text{client}
\]
Storage Area Network

- Common in large storage environments (and becoming more common)
Disk Scheduling

- The operating system is responsible for using hardware efficiently — for the disk drives, this means fast access time and large disk bandwidth.
- Access time has two major components
  - Seek time is the time for the disk area to move the heads to the cylinder containing the desired sector.
  - Rotational latency is the additional time waiting for the disk to rotate the desired sector to the disk head.
- Minimize seek time
- Seek time \( \approx \) seek distance
- Disk bandwidth is the total number of bytes transferred, divided by the total time between the first request for service and the completion of the last transfer.
Disk Scheduling (Cont.)

- Several algorithms exist to schedule the servicing of disk I/O requests.
- We illustrate them with a request queue (0-199).

98, 183, 37, 122, 14, 124, 65, 67

Head pointer 53
FCFS

Illustration shows total head movement of 640 cylinders.

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

- Selects the request currently in the request queue with the minimum seek time from the current head position.
- SSTF scheduling is a form of SJF scheduling; may cause starvation of some requests.
- While it may appear that it should minimize head movement, locally optimal decisions do not always lead to globally optimal results; in this case, SSTF may lead to substantial back-and-forth motion that will increase the head movement relative to something like SCAN or LOOK
- Illustration shows total head movement of 236 cylinders.
SSTF (Cont.)

Illustration shows total head movement of 236 cylinders.

queue = 98, 183, 37, 122, 14, 124, 65, 67

head starts at 53
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 of the disk, where the head movement is reversed and servicing continues.
- Sometimes called the *elevator algorithm*.
- The illustration on the following slide assumes that the current direction at 53 is toward smaller cylinder numbers.
SCAN (Cont.)

Illustration shows total head movement of 236 cylinders.

queue = 98, 183, 37, 122, 14, 124, 65, 67
head starts at 53
C-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.
- Treats the cylinders as a circular list that wraps around from the last cylinder to the first one.
C-SCAN (Cont.)

Illustration shows total head movement of 382 cylinders.

queue = 98, 183, 37, 122, 14, 124, 65, 67

head starts at 53
LOOK and C-LOOK

- Versions of SCAN and C-SCAN
- LOOK - Arm only goes as far as the last request in each direction, then reverses direction immediately, without continuing to the corresponding edge of the disk
- C-LOOK – Arm only goes as far as the last request towards higher cylinder numbers; head is then moved immediately to the lowest cylinder number for which there is a request
C-LOOK (Cont.)

Illustration shows total head movement of 322 cylinders.

queue  98, 183, 37, 122, 14, 124, 65, 67
head starts at 53
Selecting a Disk-Scheduling Algorithm

- SSTF is common and has a natural appeal
- SCAN and C-SCAN perform better for systems that place a heavy load on the disk.
- Performance depends on the number and types of requests.
- Requests for disk service can be influenced by the file-allocation method (why?)
- The disk-scheduling algorithm should be written as a separate module of the operating system, allowing it to be replaced with a different algorithm if necessary.
- Either SSTF or LOOK is a reasonable choice for the default algorithm.
Disk Management

- **Low-level formatting**, or *physical formatting* — Dividing a disk into sectors that the disk controller can read and write.

- To use a disk to hold files, the operating system still needs to record its own data structures on the disk.
  - *Partition* the disk into one or more groups of cylinders.
  - *Logical formatting* or “making a file system”.

- **Boot block** initializes system.
  - The bootstrap is stored in ROM.
  - *Bootstrap loader* program.

- **Methods** such as *sector sparing* used to handle bad blocks.
Swap-Space Management

- Swap-space — Virtual memory uses disk space as an extension of main memory.
- Swap-space can be carved out of the normal file system, or, more commonly, it can be in a separate disk partition.
- Swap-space management
  - 4.3BSD allocates swap space when a process starts; holds text segment (the program) and data segment.
  - Kernel uses swap maps to track swap-space use.
  - Solaris 2 allocates swap space only when a page is forced out of physical memory, not when the virtual memory page is first created.
RAID Structure

- **RAID** – multiple disk drives provides **reliability** via **redundancy**.

- RAID is arranged into six different levels.

- Several improvements in disk-use techniques involve the use of multiple disks working cooperatively.

- Disk striping uses a group of disks as one storage unit.

- RAID schemes improve performance and improve the reliability of the storage system by storing redundant data.
  - *Mirroring* or *shadowing* keeps duplicate of each disk.
  - *Block interleaved parity* uses much less redundancy.
RAID Levels

(a) RAID 0: non-redundant striping.

(b) RAID 1: mirrored disks.

(c) RAID 2: memory-style error-correcting codes.

(d) RAID 3: bit-interleaved parity.

(e) RAID 4: block-interleaved parity.

(f) RAID 5: block-interleaved distributed parity.

(g) RAID 6: P + Q redundancy.
Stable-Storage Implementation

- Write-ahead log scheme requires stable storage.

- To implement stable storage:
  - Replicate information on two or more non-volatile storage media with independent failure modes.
  - Update information in a controlled manner to ensure that we can recover the stable data after any failure during data transfer or recovery.
Tertiary Storage Devices

- Low cost is the defining characteristic of tertiary storage.

- Generally, tertiary storage is built using removable media

- Common examples of removable media are floppy disks and CD-ROMs; other types are available.
Tapes

- Compared to a disk, a tape is less expensive and holds more data, but random access is much slower.
- Tape is an economical medium for purposes that do not require fast random access, e.g., backup copies of disk data, holding huge volumes of data.
- Large tape installations typically use robotic tape changers that move tapes between tape drives and storage slots in a tape library.
  - stacker – library that holds a few tapes
  - silo – library that holds thousands of tapes
- A disk-resident file can be *archived* to tape for low cost storage; the computer can *stage* it back into disk storage for active use.
Operating System Issues

- Major OS jobs are to manage physical devices and to present a virtual machine abstraction to applications.

- For hard disks, the OS provides two abstraction:
  - Raw device – an array of data blocks.
  - File system – the OS queues and schedules the interleaved requests from several applications.
Application Interface

- Most OSs handle removable disks almost exactly like fixed disks — a new thumb drive is formatted and an empty file system is generated on the disk.
- Tapes are presented as a raw storage medium, i.e., an application does not open a file on the tape, it opens the whole tape drive as a raw device.
- Usually the tape drive is reserved for the exclusive use of that application.
- Since the OS does not provide file system services on a tape, the application must decide how to use the array of blocks.
- Since every application makes up its own rules for how to organize a tape, a tape full of data can generally only be used by the program that created it.
Tape Drives

- The basic operations for a tape drive differ from those of a disk drive.
- **locate** positions the tape to a specific logical block, not an entire track (corresponds to **seek**).
- The **read position** operation returns the logical block number corresponding to the position of the tape head.
- The **space** operation enables relative motion.
- Tape drives are “append-only” devices; updating a block in the middle of the tape also effectively erases everything beyond that block.
- An EOT mark is placed after a block that is written.
File Naming

- The issue of naming files on removable media is especially difficult when we want to write data on a removable cartridge on one computer, and then use the cartridge in another computer.
- Contemporary OSs generally leave the name space problem unsolved for removable media, and depend on applications and users to figure out how to access and interpret the data.
- Some kinds of removable media (e.g., CDs) are so well standardized that all computers use them the same way.
Hierarchical Storage Management (HSM)

- A hierarchical storage system extends the storage hierarchy beyond primary memory and secondary storage to incorporate tertiary storage — usually implemented as a jukebox of tapes or removable disks.
- Usually incorporate tertiary storage by extending the file system.
  - Small and frequently used files remain on disk.
  - Large, old, inactive files are archived to the jukebox.
- HSM is usually found in supercomputing centers and other large installations that have enormous volumes of data.
Speed

- Two aspects of speed in tertiary storage are bandwidth and latency.

- Bandwidth is measured in bytes per second.
  - Sustained bandwidth – average data rate during a large transfer; # of bytes/transfer time. Data rate when the data stream is actually flowing.
  - Effective bandwidth – average over the entire I/O time, including seek or locate, and cartridge switching. Drive’s overall data rate.
Speed (Cont.)

- Access latency – amount of time needed to locate data.
  - Access time for a disk – move the arm to the selected cylinder and wait for the rotational latency; < 35 milliseconds.
  - Access on tape requires winding the tape reels until the selected block reaches the tape head; tens or hundreds of seconds.
  - Generally say that random access within a tape cartridge is about a thousand times slower than random access on disk.

- The low cost of tertiary storage is a result of having many cheap cartridges share a few expensive drives.

- A removable library is best devoted to the storage of infrequently used data, because the library can only satisfy a relatively small number of I/O requests per hour.
Reliability

- A fixed disk drive is likely to be more reliable than a removable disk or tape drive.

- An optical cartridge is likely to be more reliable than a magnetic disk or tape.

- A head crash in a fixed hard disk generally destroys the data, whereas the failure of a tape drive or optical disk drive often leaves the data cartridge unharmed.
Cost

• Main memory is much more expensive than disk storage

• The cost per megabyte of hard disk storage is competitive with magnetic tape if only one tape is used per drive.

• The cheapest tape drives and the cheapest disk drives have had about the same storage capacity over the years.

• Tertiary storage gives a cost savings only when the number of cartridges is considerably larger than the number of drives.