Lecture Overview

- Mass storage devices
  - Disk scheduling
  - Disk reliability
  - Tertiary storage
  - Swap space management
  - Linux swap space management

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 Scheduling

- The OS is responsible minimizing *access time* and maximizing bandwidth for disks
- Access time has two major components
  - *Seek time* is the time for the disk are 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
- Disk bandwidth is the total number of bytes transferred, divided by the total time from the start of the request to the completion of the last transfer
- Requests are serviced immediately if the disk is not busy, if disk is busy then requests are queued
  - Disk scheduling chooses the next request to service

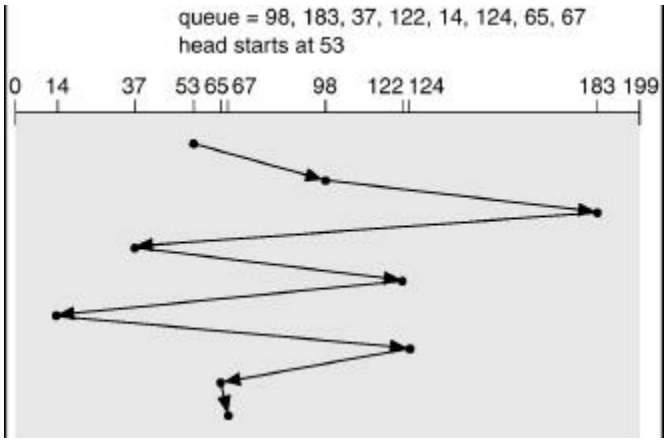
Disk Scheduling

- First-come, first-serve (FCFS)
  - As always, this is the simplest algorithm to implement
  - This algorithm is intrinsically fair, but does not necessarily provide the fastest service
  - Consider requests for blocks on the following cylinders

    98, 183, 37, 122, 14, 124, 65, 67
  - Assume the disk head is initially on cylinder 53
    - The next slide shows the head traversal

Disk Scheduling

- First-come, first-serve (FCFS)
  - Total disk head movement is 640 cylinders

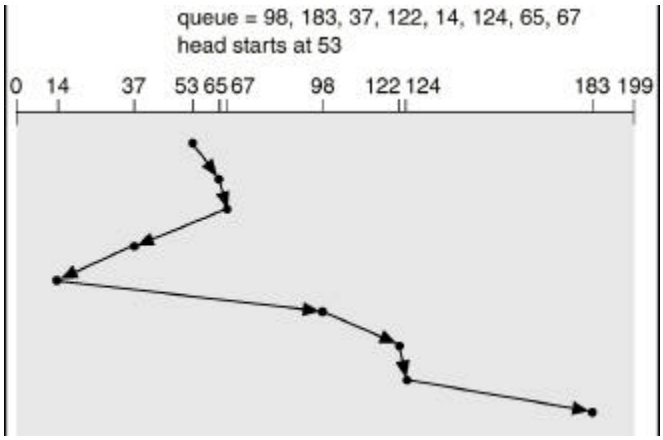


Disk Scheduling

- Shortest-seek-time-first (SSTF)
  - Service requests closest to the current head position (i.e., the minimum seek time)
  - Is a form of Shortest-job-first
  - May lead to starvation
  - Next slide depicts head movement for previous example

Disk Scheduling

- Shortest-seek-time-first (SSTF)
  - Total head movement is 236 cylinders

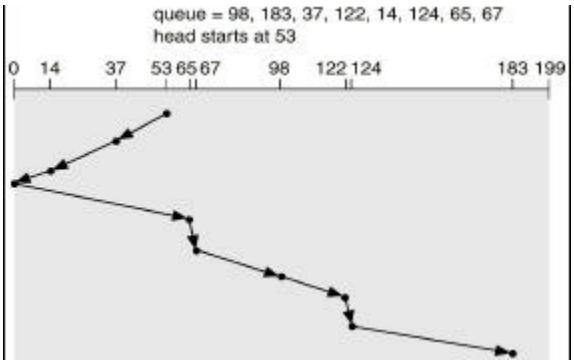


Disk Scheduling

- Elevator algorithm
  - Also called SCAN
  - The disk arm starts at one end of the disk
  - Moves toward the other end of the disk, servicing requests along the way
  - When there are no more requests or it reaches the end, it reverses direction and moves back toward the other end of the disk, servicing requests along the way
  - This process is continued repeatedly
  - Next slide depicts head movement for previous example

Disk Scheduling

- Elevator algorithm
  - Total head movement is 208 cylinders
    - This figure assumes a “strict” elevator that moves all the way to the end before reversing directions

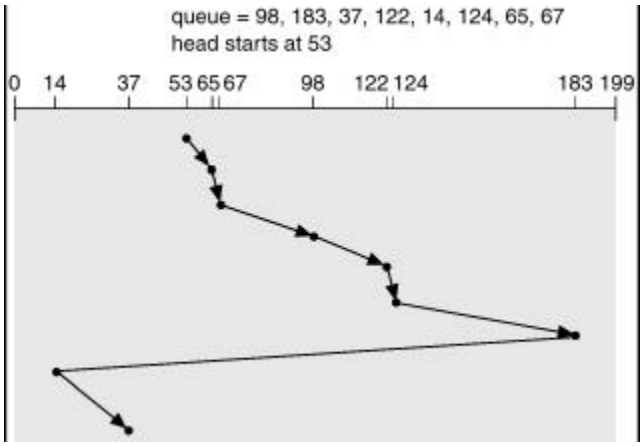


Disk Scheduling

- C-SCAN
  - “Circular” SCAN
  - Provides a more uniform wait time than Elevator
  - The head moves from one end of the disk to the other, servicing requests as it goes
  - When it reaches the other end, 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

Disk Scheduling

- C-SCAN
  - Head movement for previous example



Selecting a Disk Scheduling Algorithm

- SSTF is common and has a natural appeal
- Elevator 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
- 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 Elevator is a reasonable choice for the default algorithm

Disk Reliability

- Several improvements in disk-use techniques involve the use of multiple disks working cooperatively
- *Disk striping* uses a group of disks as one
  - Improves performance by breaking blocks into sub-blocks
- *RAID* (redundant array of independent disks) 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 Schemes

- Defined in 1988 by Patterson, Gibson, and Katz from Berkeley University
- RAID 0: Striping
- RAID 1: Mirroring
- RAID 2: Bit Striping with ECC (use Hamming-code; practically not used)
- RAID 3: Bit striping with parity
- RAID 4: Striping with fixed parity
- RAID 5: Striping with striped parity
- RAID 10: Striped mirrors

Tertiary Storage

- 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, CD-ROMs, etc.

Tertiary Storage

- Floppy disk — thin flexible disk coated with magnetic material, enclosed in a protective plastic case
  - Most floppies hold about 1 MB; similar technology is used for removable disks that hold more than 1 GB
  - Removable magnetic disks can be nearly as fast as hard disks, but they are at a greater risk of damage from exposure

Tertiary Storage

- A magneto-optic disk records data on a rigid platter coated with magnetic material
  - Laser heat is used to amplify a large, weak magnetic field to record a bit
  - Laser light is also used to read data (Kerr effect)
    - Polarization of laser is rotated depending on magnetic field orientation
  - The magneto-optic head flies much farther from the disk surface than a magnetic disk head, and the magnetic material is covered with a protective layer of plastic or glass; resistant to head crashes

Tertiary Storage

- Optical disks do not use magnetism; they employ special materials that are altered by laser light
  - Phase-change disk
    - Coated with material that can freeze into either a crystalline or an amorphous state
    - Different states reflect laser light with different strengths
    - Laser is used at different power settings to melt and refreeze spots on disk to change their state
  - Dye-polymer disk
    - Coated with plastic contained a dye that absorbs laser light
    - Laser can heat a small spot to cause it to swell and form a bump
    - Laser can reheat a bump to smooth it out
- High cost, low performance

Tertiary Storage

- Data on read/write disks can be modified many times
- *WORM* ("Write Once, Read Many Times") disks can be written only once
- Thin aluminum film sandwiched between two glass or plastic platters
- To write a bit, the drive uses a laser light to burn a small hole through the aluminum; information can be destroyed by not altered
- Very durable and reliable

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
- Swapping serves two primary purposes
  - Expands usable address space for processes
    - For example, programs that are bigger than physical RAM can still be executed
  - Expands amount of available RAM to load processes
    - For example, an entire program does not need to be in memory in order to execute

Linux Swap Space Management

- Performed at the page level
  - Requires hardware paging unit in the CPU
- Linux page table entry
  - `Present` flag indicates whether a page is swapped out
  - Remaining bits store location of page on disk
- Linux only swaps
  - Pages belonging to an anonymous memory region of a process (e.g., user mode stack)
  - Modified pages belonging to a private memory mapping of a process
  - Pages belonging to an IPC shared memory region
  - Other page types are either used by the kernel or to map files on disk and are not swapped to the swap area

Distributing Pages in a Linux Swap Area

- A swap area is organized into slots
  - A slot contains exactly one page
- When swapping out, the kernel tries to store pages in contiguous slots so as to minimize seek time
- If more than one swap area is used
  - Swap areas on faster disks get higher priority
  - When looking for a free slot, the search starts with the area with the highest priority
    - If several have the same priority then a cyclical selection is made among them to avoid overloading
  - If no free slots are found in the highest priority swap area, then the search continues down to the next highest priority

Linux Page Swap Selection

- Generally, the rule for selecting pages to swap is to take from the process with largest number of pages
  - Changes in 2.4 to processes with fewest page faults
- A choice must also be made among process pages
  - As we know, LRU is a good selection algorithm
  - The x86 processor does provide hardware support for LRU
  - Linux tries to approximate LRU by using the `Accessed` flag in each page table entry; this flag is automatically set by the hardware when the page is accessed

When to Swap in Linux

- Linux swaps out pages on the following occasions
  - When the number of free page frames falls below a predefined threshold, the `kswapd` thread is activated once every second to swap pages
  - When a request to the Buddy system cannot be satisfied because the number of free frames would fall below a predefined threshold

Linux Swap Area

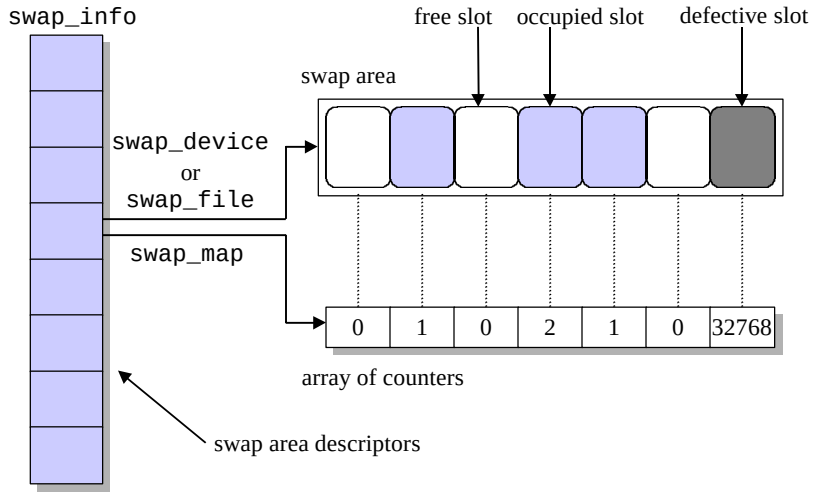
- Linux can support up to `MAX_SWAPFILES` swap areas (usually set to 8)
- Each swap area consists of a sequences of page slots, i.e., 4096 byte blocks
- The first slot of the swap area is used to store information about the swap area
  - The first slot is a `swap_header` union containing two structures
    - magic - contains magic number to identify partition as swap stored at end of slot
    - info - contains information such as swap algorithm version, last usable page slot, number of bad page slots, and location of bad page slots

Linux Swap Area

- Each swap area also has an in-memory descriptor of type `swap_info_struct` stored in the `swap_info` array
  - Contains device number, dentry of file or device file, priority, pointer to an array of counters (one counter for each slot), pointer to an array of bit locks for each swap area, last slot scanned, next slot to scan, etc.
    - Counters are 0 if slot is free, positive value if not free, `SWAP_MAP_MAX` (32767) if slot is permanent, `SWAP_MAP_BAD` (32768) if slot is defective
  - Descriptors also form a list sorted by priority

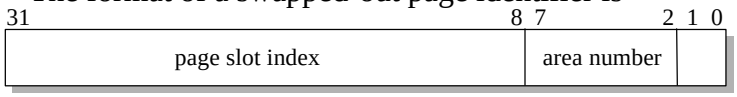
Linux Swap Area

- Swap area data structures



Swapped-Out Page Identifier

- A swapped-out page is uniquely identified by specifying the index of the swap area in the `swap_info` array and its slot index
  - The first possible slot index is 1, since 0 is used
  - The format of a swapped-out page identifier is



- This identifier is inserted into page table when a page is swapped out, table entry has three cases
  - Null entry means page is not in process address space
  - Page swapped out if least two bits are zero and 30 remaining are non-zero
  - Least bit is 1, then page is in memory

Finding a Free Page Slot

- The goal is to store pages in contiguous slots
- Allocating from the start of the swap area increases average swap-out time, while allocating from the last allocated slot increases swap-in time
  - Linux' approach tries to balance these issues
- Linux always searches for free slots from the last allocated slot, unless
  - The end of the swap area is reached
  - The number of allocated slots since the last restart is greater than or equal to `SWAPFILE_CLUSTER` (usually 256)

The Swap Cache

- Linux allows page frames to be shared when
  - The frame is associated with shared memory mapping
  - The frame is used for copy-on-write operation
  - The frame is allocated to an IPC shared memory resource
- The swap area is not used in the first case (since the mapped file is used instead)
- It is possible for a page to be swapped out from one process, but still be in use in another process that is sharing the page
  - The swap cache collects all shared page frames that have been copied to swap areas

The Swap Cache

- It is possible that a page will get swapped out multiple times, which is why the slot counter in swap descriptor may have a value greater than 1
  - The counter is incremented each time a page is swapped out
- It is possible to swap out a shared page for all process at once, but it is inefficient since the kernel does not know which processes share the page

The Swap Cache

- Consider page P shared among processes A and B
  - If P is selected for swapping in process A, a slot is allocated for P and the data for P is copied to the slot
  - The swapped-out identifier is put in process A's page table
  - The page's usage counter does not become zero, since process B is still using the page
    - Thus the page was swapped, but not freed
  - Now suppose P is selected for swapping from process B
  - It first checks to see if P is in the swap cache (i.e., it has already been swapped out), if so it gets its swapped-out identifier and puts that in process B's page table
  - The frame will be freed when all processes sharing the page have swapped it out
  - A similar cache look-up is needed for bring the page back in