How to organize data on disks.

Brief review of how disks work.

Disk management

Today: File System Implementation

Last Class: File System Abstraction
Bandwidth: once a transfer is initiated, the rate of I/O transfer

- Rotational time: the time for the correct sector to rotate under the head
- Seek time: time to position the head over the correct cylinder

Latency: the time to initiate a disk transfer of 1 byte to memory

Overhead: time the CPU takes to start a disk operation

Disk Overheads

Each track is split into sectors or blocks, the minimum unit of transfer from the disk
Tracks are concentric rings on disk with bits laid out serially on tracks.

Spinning (like a CD).

The disk surface is circular and is coated with a magnetic material. The disk is always

How Disks Work
3. How do we lay out the files on the physical disk?

2. What is the right data structure in which to maintain file location?

I. We need to support sequential and random access.

Key performance issues:

- Platter 4, cylinder 3, sector 8
- Platter 0, cylinder 0, sector 0

The information we need:

File Organization on Disk

How disks work:

- Select and transfer the correct sector as it spins by.
- Move arm to correct track, waiting for the disk to rotate under the head.
- Disk operations are in terms of radial coordinates.
  - Cylinders are matching sectors on each surface
  - Comb has 2 read/write head assemblies at the end of each arm.
  - Disk packs use both sides of the platters, except on the ends
  - Platters
- CDs come individually, but disks come organized in disk pack consisting of a stack of
Examples: IBM OS/360, write-only disks, early personal computers

- Fragmentation? Disk management?
  - Changing file sizes

Disadvantages

- Access time? Number of seeks?
  - Simple?

Advantages

- Need to store only the start location and size in the file descriptor
- OS allocates a contiguous chunk of free blocks when it creates a file.
- OS maintains an ordered list of free disk blocks

Continuous Allocation

...
Examples: MS-DOS
- Number of seeks?
- Does not support which type of access? Why?

Disadvantages:
- Efficiency supports which type of access?
- File size changes?
- Fragmentation?

Advantages:

linked files

In each sector, keep a pointer to the next sector.

In the file descriptor, keep a pointer to the first sector/block.

Keep a list of all the free sectors/blocks.
Examples: Nachos

- Lots of seeks because data is not contiguous.
- Sets a maximum file size.

Disadvantages

- Both sequential and random accesses are easy.
- Not much wasted space.

Advantages

Indexed Files

Descriptor

File

OS files in the pointers as it allocates blocks.

OS creates the file, but allocates the blocks only on demand.

OS allocates an array to hold the pointers to all the blocks when it is created.

The user of OS must declare the maximum length of the file when it is created.

OS keeps an array of block pointers for each file.

Indexed Files
What could the OS do to get more continuous access and fewer seeks?

- Is the file size bounded?

  - Lots of seeks because data is not continuous.
  - Indirect access is inefficient for random access to very large files.

**Disadvantages**

- Small files?
  - Support incremental file growth
  - Simple to implement

**Advantages**

---

**Multilevel indexed files: BSD UNIX 4.3**

---

---

**Multilevel indexed files**

- 14th pointer points to a block of pointers to index blocks. (Two indirects)
- 13th pointer points to a block of 1024 pointers to 1024 more data blocks. (One
  indirect)
- First 12 pointers point to data blocks.
- Each file descriptor contains 14 block pointers.
Bitmaps are implemented in userProc/Bitmap

Bitmaps in Naches

Marking a block as free is simple since the block number can be used to

1100010100011111110

... operations to find an empty block.

- If the bit is 1, the block is free.
- If the bit is 0, the block is allocated.

- The bitmap has one bit for each block on the disk.
- Need to be able to find free space quickly and release space quickly

We need a free-space list for main memory.

Free-Space Management
How expensive is it to allocate a block?

How expensive is it to free a block?

Next free block.

The head of the list is cached in kernel memory. Each block contains a pointer to the next free block.

An alternative implementation is to link together the free blocks.

Bitmap:

If most of the disk is in use, it will be expensive to find free blocks with a

(500,000 bytes).

2 GB disk with 512 byte sectors requires a bitmap with 4,000,000 entries

Problem: Bitmap might be too big to keep in memory for a large disk.

Free-Space Management

Free-Space Management

A more efficient implementation of contiguous allocation could be

accomplished by adding a method to Bitmap to find a contiguous chunk.

A more efficient implementation of contiguous allocation could be

- If a contiguous block is found, use a second loop to set the blocks by calling Mark.

  - Check each block.

- If so, loop to find a contiguous number of blocks meeting your need, using Find.

  - Rest to exist.

To allocate a contiguous block, use NumClear to see if enough free blocks

and return its index. Returns -1 if none are available.

To allocate an item, call Find. It will search for a free item, set its bit,

To free an item, call Clear. Pass the index for the item.

the items are cleared.

Create a Bitmap for the number of items you want to track. Initially, all

Using Bitmaps in Nachos
2. Memory allocation strategies:

I. What is virtual memory and why do we use it?

Topics you should understand:

Memory Management

Exam Review

Summary

- Free space can be managed using a bitmap or a linked list.
  - to physical disk blocks;
- Indexed allocation is very similar to page tables. A table maps from logical file blocks to physical disk blocks and needs to move files as they grow.
- Continuous allocation is simple, but suffers from external fragmentation. The need for implementations are similar to those of virtual memory implementations.
  - Many of the concerns and implementations of the system...
corresponding physical address.

- Given a virtual address and the necessary tables, determine the
  corresponding physical address.

- Using contiguous allocation:
  - Given a request for memory, determine how the request can be satisfied.

Things you should be able to do:

**Memory Management (cont.)**

What needs to happen on a context switch to support memory management:

- Memory protection
- Ability to move processes
- Ability to share memory with other processes
- Ability to grow processes
- Coping with fragmentation
- Hardware support required
- Address translation

For each strategy, understand these concepts:

**Memory Management (cont.)**
What is a working set?

What is the performance of paging?

What is temporal locality? What is spatial locality? What effect do these have on the performance of paging?

How do global and per-process (aka local) allocation differ?

Page replacement algorithms. For each understand how they work.

Page (cont.)

- Enhanced second chance
- Second chance
- LRU
- MIN
- FIFO

Advantages, disadvantages, and hardware requirements.

What does the OS do when a page fault occurs?

What is a page fault, how does the OS know it needs to take one, and what is demand paging?

What is a TLB? How is one used?

What does the OS store in the page table?

What is paging? A page, a page frame?

Topics you should understand:

Page
(c) Indexed
(b) Linked
(a) Continuous allocation

8. Strategies for laying files out on disk: Advantages and disadvantages.
7. How does Unix support multiple users of shared files?
6. What happens if the directory structure is a graph?
5. What is a link?
4. What is a directory?
3. How does the O5 do on a file open, file close?
2. What types of access are typical for files?
1. What is a file, a file type?

**Topics you should understand:**

**File Systems**

---

**Page Faults.**

determine how the different replacement algorithms would handle the

I. Given a page reference string and a fixed number of page frames,

Things you should be able to do:

- What considerations influence the page size that should be used?

- What is thrashing and what are strategies to eliminate it?

**Paging (cont.)**
... system, such as Unix, Nacos, Windows NT, ...

- You will **not** be asked detailed questions about any specific operating system, such as Unix, Nacos, Windows NT, ...
- You will **not** be asked to read or write C++ code.
- Changes in one part of the OS might impact another.
- You should have a good sense of how the pieces fit together and how

General Skills