Storage Hardware

Owen.Conlan@cs.tcd.ie
Hardware Issues

Primary Storage

CPU → Cache → Main Memory

Secondary Storage

- Hard Disk/SSD
- CD ROM/RW
- DVD ROM/RW
- Tapes
- Floppy Disk/Memory Stick
Hardware Issues

Primary Storage is …

- Limited
- Volatile
- Expensive

However, it is also …

- Fast (May be accessed directly from the CPU)
Hardware Issues

Secondary Storage is …

• Extendible
• Persistent
• Cheap

However, it is …

• Relatively slow (must be copied to main memory before being accessed by the CPU)
Hardware Issues

Why do we use secondary storage?

• Primary storage (RAM) costs more than disk space
• We like to switch our computers off and on again

But secondary storage is very slow

• Retrieving a single character from RAM takes about 150 nanoseconds (150 billionths of a second)
• Retrieving the same character from disk takes about 75 milliseconds (thousandths of a second)
• 75 msec is 500,000 times longer than 150 ns.
Disk Storage Devices

Disk Storage Devices …

• Direct Access Storage
  – As opposed to Tape drives, which are serial devices
• Offer high storage capacity and low cost
• Data stored as magnetized areas on magnetic platters surfaces
• Each disk has one or more platters
• A disk pack contains several magnetic platters connected to a rotating spindle
Disk Pack with Read/Write H/W

- Actuator
- Arm
- Read/Write Head
- Spindle
- Cylinder of Tracks (imaginary)
- Magnetic Disk
- Actuator Movement
- Rotation
Movable vs Fixed-head Disks

Some disks have fixed-heads

- As many read/write heads as there are tracks on the platter
- Track is selected electronically and is therefore much faster
- Cost of additional read/write heads is the limiting factor to production

Disks with an actuator are called moveable-head disks

- Actuator moves the (single) read/write head per platter to the appropriate track
Tracks and Sectors

- Disks are divided into concentric circular tracks on each platter surface.
- Track capacities vary typically from 4 to 50 Kbytes.
- The division of the disk into sectors is hard-coded and may not be changed.
- Subtended at fixed angle at the centre of a platter are sectors (figure above).
- Not all disks have their tracks divided into sectors.
Cylinder

Tracks directly above and below one another form a cylinder.

All information on a cylinder can be accessed without moving the arm (called seeking) that holds the read/write heads.

- A cylinder consists of a group of tracks
- A track consists of a group of sectors
- A sector consists of a group of bytes
Estimating Capacities

Track capacity = #sectors per track * bytes per sector
Cylinder capacity = #tracks per cylinder * Track capacity
Drive capacity = #cylinders * Cylinder capacity

Knowing these relationships allows us to compute the amount of disk space a file is likely to require

• How many cylinders to store a file with 20,000 fixed length records of 256 bytes each on a disk with –
  – 512 bytes per sector
  – 40 sectors per disk
  – 11 tracks per cylinder
Estimating Capacities

Answer –

- The file is $20,000 \times 256 = 5,120,000$ bytes
- $5,120,000 / 512 = 10,000$ sectors
- $10,000 / 40 = 250$ tracks
- $250 / 11 = 22.72$ cylinders

If 22.72 physically contiguous cylinders are not available then the file will have to be spread out over the disk (fragmentation)
**Blocks**

What are Blocks?

- A track is divided into **blocks** or **pages**
- Block size (generally) fixed for each operating system
- Typical block sizes range from 512 bytes to 4096 bytes
- A disk with hard-coded sectors often has the sectors further subdivided onto blocks

Why are they important?

- Whole blocks are transferred between disk and main memory for processing
Input/Output (I/O) from/to a Disk …

- A read-write head moves to the track that contains the block to be transferred (seek)
- Disk rotation moves the block under the read-write head for reading and writing (rotational delay or latency)
- Operating System finds the correct block
  - A physical disk block address consists of a surface number, track number (within surface), and block number (within track)
- Entire block read/written from/to an area in RAM called a buffer
- Time taken to transfer the block (block transfer time)

Reading or writing a disk block is time consuming because of the seek times and rotational delay

- Double buffering can be used to speed up the transfer of contiguous disk blocks
Buffers

When several blocks need to be transferred from disk to main memory (and all the block addresses are known beforehand) several buffers can be reserved in main memory to speed the transfer

- CPU processes data in a buffer while another is being read/written

Facilitated by independent Input/Output processor

- Can transfer data block to/from main memory independent of and in parallel to CPU processor
- This technique is referred to as double buffering
Example of Double Buffering

Use of two buffers, A and B, for reading from disk

\[
\begin{array}{c|c|c|c|c|}
\text{Disk Block} &: & i & i + 1 & i + 2 & i + 3 & i + 4 \\
& & \text{fill A} & \text{fill B} & \text{fill A} & \text{fill B} & \text{fill A} \\
\hline
\text{IO} & \rightarrow & \rightarrow & \rightarrow & \rightarrow & \rightarrow \\
\hline
\text{Disk Block} &: & i & i + 1 & i + 2 & i + 3 & i + 4 \\
& & \text{process} A & \text{process} B & \text{process} A & \text{process} B & \text{process} A \\
\hline
\text{Processing} & \rightarrow & \rightarrow & \rightarrow & \rightarrow & \rightarrow \\
\hline
\end{array}
\]
Physical and Logical Files

A disk drive may contain many physical files

• A program uses these files via the operating system, but knows nothing about where the data is on disk

A program sees only *logical files*

• When a program wants to access a file, the operating system provides the hook-up between logical and physical file
• The programmer does not need to know the physical location on disk, just the *logical* name creating
There are two basic ways to organise data on a disk – by **sector organisation** or **user-defined block**

We will study only sector organisation

The physical placement of sectors –

- You cannot read a series of sectors that are all in the same track one right after another!
- After reading the data, it takes the machine time to process the data before retrieving more
- So, if logically adjacent sectors were placed **physically** adjacent, retrieval would be slow
- Solution: **Interleave** the sectors
Sector Organisation - Clusters

A cluster is a fixed number of contiguous sectors

- All clusters on a disk are the same size
- To view a file as a series of clusters and still maintain the sectored view, the file manager ties the logical sectors to the physical clusters using a file allocation table (FAT)

FAT -

- The FAT contains a linked list of all the clusters in a file, ordered according to the logical order of the sectors in a cluster
- With each entry in the FAT is an entry giving the physical location of the cluster
Review

Compared to RAM disk accesses are slow, but cheap!
Disks facilitate persistent storage of data
Disks enable direct access

Anatomy of a Disk
- Platters, tracks, cylinders, sectors, blocks

Physical components of a Disk
- Magnetic disks, actuator, arm, read/write heads, spindle

Estimating Capacities

Disk I/O and Buffering

Sector Organisation and Clustering
File Basics

Owen.Conlan@cs.tcd.ie
Files

Data placed on secondary storage is collected into files.

The layout of the data in the file will have an enormous impact on how long it takes to find a particular piece of data.

We don’t want to have to look through the whole file if we are looking for one small piece of data.

We want to find the right piece of data with as few accesses to the file as possible.
Logical Files

Program

Logical Files

Operating System

Physical Files

Other Hardware Devices
Consider the following information –

- Lord of the Rings, 189, Fantasy
- Star Wars, 210, Sci-Fi

When this information is written to a (stream) file, we lose the organisation of the units of data.

When the file is accessed by a program its information is seen only as a string of characters.

Solution: Use records with field structures.
wilde> od -A x -xc lotr.txt
000000 4c6f 7264 206f 6620 7468 6520 5269 6e67
  Lord of the Ring
000010 732c 2031 3839 2c20 4661 6e74 6173 790a
   s, 189, Fantasy
000020

od is a program which enables the user to dump file information in octal and other formats.

The output above is shown as hexidecimal shorts.
What is a **Field**?

- A **Fixed** or **variable** number of bytes that form a data value
- E.g. Movie Name such as Star Wars

There are many ways of adding structure to files to maintain the identity of fields, for example

- Choose a special character/delimiter that will not appear as a legitimate character within a field and then insert that character into the file after writing each field… called **delimited-text field**
- Use a fixed length for each field (the size depending on field in question) and pad out when length of actual data value is less than the fixed length… called **fixed-length field**
- Write the length of the value (in bytes) followed by the value in exactly that number of bytes… called **length-based field**
- Write the name of the field and then value both represented as delimited-text fields… called **identified field**
What is a **Record**?

- Fixed or variable length collections of fields
- E.g. Movie Record such as Star Wars, 210, SciFi

A record is a set of fields that belong together when the file is viewed in terms of a higher level organisation. A record is another level of organisation that we impose on the data to preserve meaning. Records do not necessarily exist in the file in any physical sense. But they are an important logical notion.
Records to Blocks mapping

Records of a file must be allocated to disk blocks

• Recall, a block is the unit of data transfer between disk and memory

Blocking Factor (bfr) for a file

• Number of records that can fit in a block when block size > record size
• bfr = round_down(B/R)
  – B is block size in bytes
  – R is actual record size in bytes for fixed length records but is the average record size in bytes for variable length records
• Unused portion = B – (bfr*R) bytes
Spanned/Unspanned Organisation

What happens in case $R > B$?

- **Spanned organisation**
  - *End of first block points to rest of record in an unused portion of another block*

- **Unspanned organisation**
  - *Each record starts at a new block and uses next block, with unused portions not used*
Files of Records

- A file is a **sequence** of records, where each record is a collection of data values (fields or data items).
- A **file descriptor** (or file header) includes information that describes the file, such as the **field names** and their **data types**, and the addresses of the file blocks on disk.
- Records are stored on disk blocks.
- The **blocking factor** for a file is the (average) number of file records stored in a disk block.
- The physical disk blocks that are allocated to hold the records of a file can be **contiguous**, **linked**, or **indexed**.
- In a file of fixed-length records, all records have the same format.
  - Usually, **unspanned blocking is used with such files**.
- Files of variable-length records require additional information to be stored in each record, such as **separator characters** and **field types**.
  - Usually **spanned blocking is used with such files**.
File Operations

**OPEN**: Readies the file for access, and associates a pointer that refers to a *current* file record at each point in time

**FIND**: Searches for the first file record that satisfies a certain condition, and makes it the current file record

**FINDNEXT**: Searches for the next file record (from the current record) that satisfies a certain condition, and makes it the current file record
File Operations

**READ**: Reads the current file record into a program variable

**INSERT**: Inserts a new record into the file, and makes it the current file record

**DELETE**: Removes the current file record from the file, usually by marking the record to indicate that it is no longer valid

**MODIFY**: Changes the values of some fields of the current file record
File Operations

CLOSE: Terminates access to the file

REORGANISE: Reorganises the file records. For example, the records marked deleted are physically removed from the file or a new organisation of the file records is created.

READ_ORDERED: Reads the file blocks in order of a specific field of the file
Review

Operating Systems provide a logical view of physical files to Programs

Stream Files

Fixed and Variable Fields
  • Delimited-text, fixed-length, length-based, identified

Records (and their relationship to Blocks)
  • bfr, spanned/unspanned

Files of Records

File Operations