Chapter 5 Managing Files and Records

File Structures
Other Ideas
Portability & Standardization

Access Methods

Sequential: Move through the file starting at the beginning until you find the record you are looking for.

- This is O(n) so it is very slow,
  Suppose you have 600,000 records on average you will look at

  (600,000/2) = 300,000 records

- If the file is sorted you don't have to look at everything before you know you won't find the record

- This really isn't a lot of help though

Direct Access: If you know the sector or track of the record or even its relative position from the front of the file you can go directly to it.

- Uses indices and/or hashing (more on this later)
- Would like $O(1)$ or close

Both Direct Access and Sequential Access require a Primary Key.

Primary Keys: Associated with each record. They must be

- data-less
- unchanging
- unambiguous
- unique

- So a primary key is some value that uniquely identifies the record

- A secondary key may identify one or more records. They don't have to be

  - data-less
  - unambiguous
  - unique
  - sometimes they don't have to be unchanging

Header Records

Most files contain a **header** record that contains information about the other information in the file.

What might be in a header record

- SIZE of header record
- NUMBER of data records
- SIZE of data records
- BLOCKING factor

along with any other useful information

The BUFFER FILE classes must take into account and use this record.

Some files don't even have records in them.

Header records for these files have identifying information about the data.

- Is it executable
- Is it a WORD file
- Is it an EXCEL file
- Is it an object file
- Is it a picture, if so, what is the format

That is, which program can be used to make sense of this file.

The OS looks at the header record to make the decision about which program to open the file with.

PORTABILITY AND STANDARDIZATION

Sharing files across platforms

Operation Systems, DOS, VMS, UNIX, Windows, Linux

Languages, C, C++, Pascal, Ada, COBOL, JAVA

Machine Architecture, SUN, IBM, PC's, VAX

Read for yourself how this is done

PORTABLE: a product is portable if it is significantly easier to modify the product as a whole in order to run it on another compiler/hardware/OS configuration than to recode it from scratch

REUSABLE: you can take components of one product in order to facilitate development of a different product with different functionality

INTEROPERABILITY: the mutual cooperation of object code from different venders, written in different languages and running on different platforms

PORTABILITY: How to

- Isolate any necessary implementation dependent pieces

  Example: Original UNIX (1978)

  9000 lines of C

  1000 lines of Assembler (Kernal)

  - Kernal must be rewritten for each implementation
  - ~ 1000 lines of C code is device drivers which must be rewritten each time the hardware changes
  - 8000 remaining lines do not have to be rewritten
- Use levels of abstraction
- Use high level languages & use the ANSII standard version
- Use a popular OS for the type of application

  POSIX – Standard interface between applications & UNIX
- Use coding standards
- Isolate the use of nonstandard language features
- Localize all OS system calls

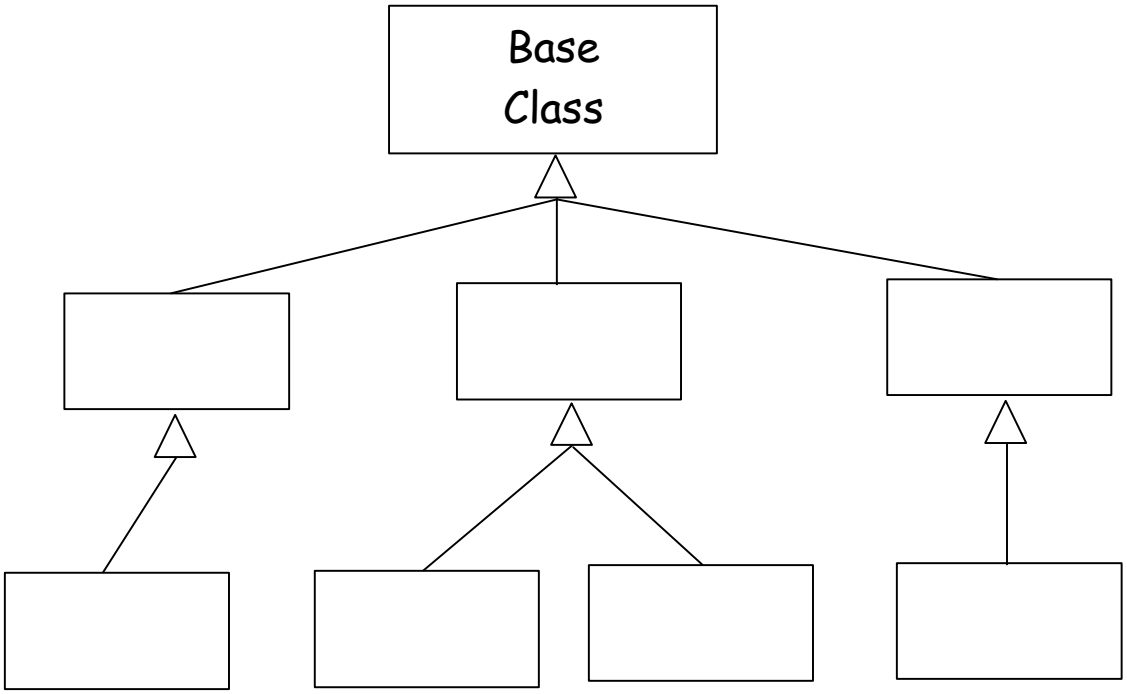
DATA PORTABILITY

- This is not easy to do
- Create unstructured (sequential file)
- Use this file to create a new data-base

CLASSES AND REUSIBILITY

- If we design our classes well with reusability in mind:
  - maintenance costs are lower (18%)
  - software development costs are lower (0%)
- What gets reused most often is short segments of code
- Major cost savings in maintenance

- Problems:Inheritance



- Any change to the base class requires testing of all the derived classes

FRAGILE BASE CLASS PROBLEM

- When we change the base class
  - derived classes must be recompiled
  - methods may need t be recoded

- Size of derived objects
  - Derived objects inherit all attributes of base class unless this is explicitly prevented
  - These objects get huge
  - So, use inheritance carefully and only where appropriate

- Keep the data private and create protected routines to change the data. If you make these **`inline`** they will not require a function call

  Now if you change the data you do have to recompile the derived classes but you don't have to change code in the derived classes

INTEROPERABILITY

Example: ATM's

C++ on the ATM

COBOL on the data base

Microsoft

OLE – Object Linking & Embedding Language that allows us to put spread sheets and pictures into word

COBRA – Common Object Request Broker Architecture Supports interoperability across vendors and machines in a distributed environment

In all of these we have systems where different groups agree to do things a certain way. This predictability makes it possible for the software to work "correctly".