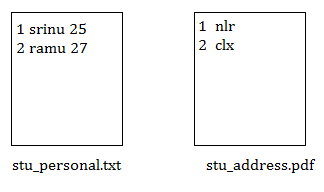
**File System VS DBMS**:

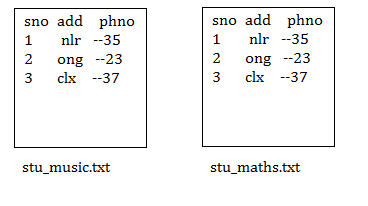
This typical file-processing system is supported by a conventional operating system. The system stores permanent records in various files, and it needs different application programs to extract records from, and add records to, the appropriate files. The application programs are written in c ,C++ ,Java …etc. Before database management systems (DBMSs) were introduced, organizations usually stored information in such systems. Keeping organizational information in a fileprocessing system has a number of major disadvantages:

A)Data isolation:-Because data are scattered in various files, and files may be in different formats, writing new application programs to retrieve the appropriate data is difficult.

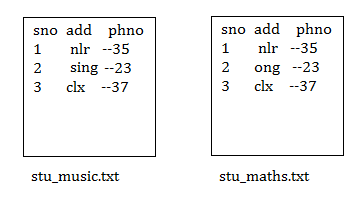


We should write a program(student.c) to get student data from above two files.Writing student.c is difficult.

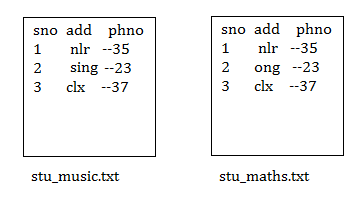
B)Data Redudancy&Inconsistency:- The same information may be duplicated in several places (files). For example, if a student has a double major (say, music and mathematics) the address and telephone number of that student may appear in a file that consists of student records of students in the Music department and in a file that consists of student records of students in the Mathematics department. This redundancy leads to higher storage and access cost.



In addition, it may lead to data inconsistency; that is, the various copies of the same data may no longer agree. For example, a changed student address may be reflected in the Music department records but not elsewhere in the system(maths department).



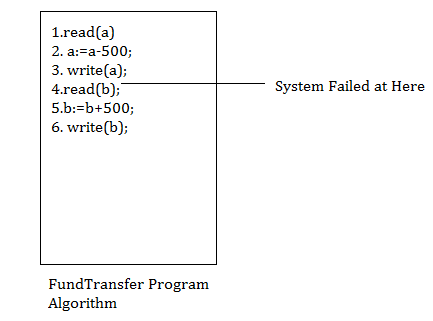
3. Difficulty in accessing data:-Suppose that one of the university clerks needs to find out the names of all students who live within a particular town. The clerk asks the data-processing department to generate such a list. Because the designers of the original system did not anticipate this request, there is no application program on hand to meet it. New application program should be developed to generate the list of all students.



4. Atomicity problems:-A computer system, like any other device, is subject to failure. In many applications, it is crucial that, if a failure occurs, the data be restored to the consistent state that existed prior to the failure.

Consider a program to transfer $500 from the account balance of department A to the account balance of department B. If a system failure occurs during the execution of the program, it is possible that the $500 was removed from the balance of department A but was not credited to the balance of department B, resulting in an inconsistent database state. Clearly, it is essential to database consistency that either both the credit and debit occur, or that neither occur.

That is, the funds transfer must be atomic—it must happen in its entirety or not at all. It is difficult to ensure atomicity in a conventional file-processing system.



5**.Integrity problems**:-The data values stored in the database must satisfy certain types of consistency constraints. Suppose the university maintains an account for each department, and records the balance amount in each account. Suppose also that the university requires that the account balance of a department may never fall below zero. Developers enforce these constraints in the system by adding appropriate code in the various application programs. However, when new constraints are added, it is difficult to change the programs to enforce them. The problem is compounded when constraints involve several data items from different files.