Title:

Department maintains a student information. The file contains roll number, name, division and address. Allow user to add, delete information of student. Display information of particular employee. If record of student does not exist an appropriate message is displayed. If it is, then the system displays the student details. Use sequential file to main the data.

Objectives:

1. To understand concept of file organization in data structure.
2. To understand concept & features of sequential file organization.

Learning Objectives:

- ✓ To understand concept of file organization in data structure.
- ✓ To understand concept & features of sequential file organization.

Learning Outcome:

- Define class for sequential file using Object Oriented features.
- Analyze working of various operations on sequential file .

Software Required: g++ / gcc compiler- / 64 bit Fedora, eclipse IDE

Theory:

File organization refers to the relationship of the key of the record to the physical locationof that record in the computer file. File organization may be either physical file or a logical file.A physical file is a physical unit, such as magnetic tape or a disk. A logical file on the other handis a complete set of records for a specific application or purpose. A logical file may occupy a partof physical file or may extend over more than one physical file.

There are various methods of file organizations. These methods may be efficient for certain types of access/selection meanwhile it will turn inefficient for other selections. Hence it isup to the programmer to decide the best suited file organization method depending on his requirement.

Some of the file organizations are

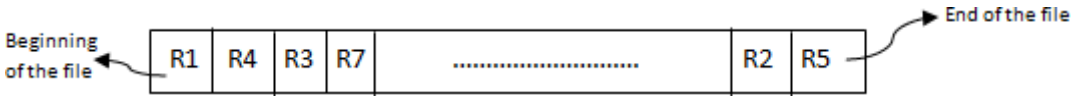
1. Sequential File Organization

2. Heap File Organization
3. Hash/Direct File Organization
4. Indexed Sequential Access Method
5. B+ Tree File Organization
6. Cluster File Organization

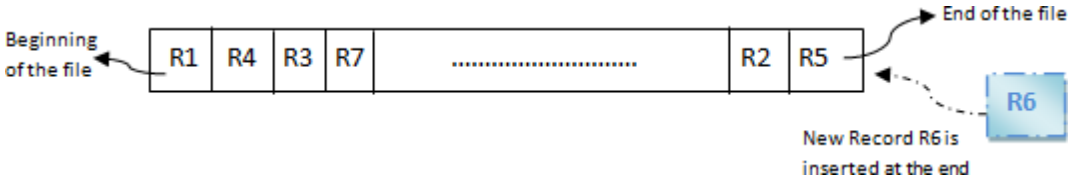
Sequential File Organization:

It is one of the simple methods of file organization. Here each file/records are stored oneafter the other in a sequential manner. This can be achieved in two ways:

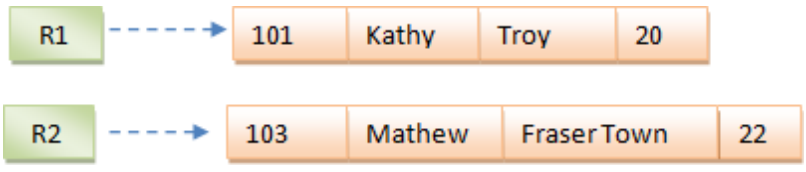
- Records are stored one after the other as they are inserted into the tables. This method is called pile file method. When a new record is inserted, it is placed at the end of the file.In the case of any modification or deletion of record, the record will be searched in the memory blocks. Once it is found, it will be marked for deleting and new block of recordis entered.



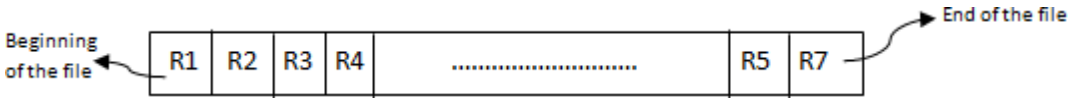
Inserting a new record:



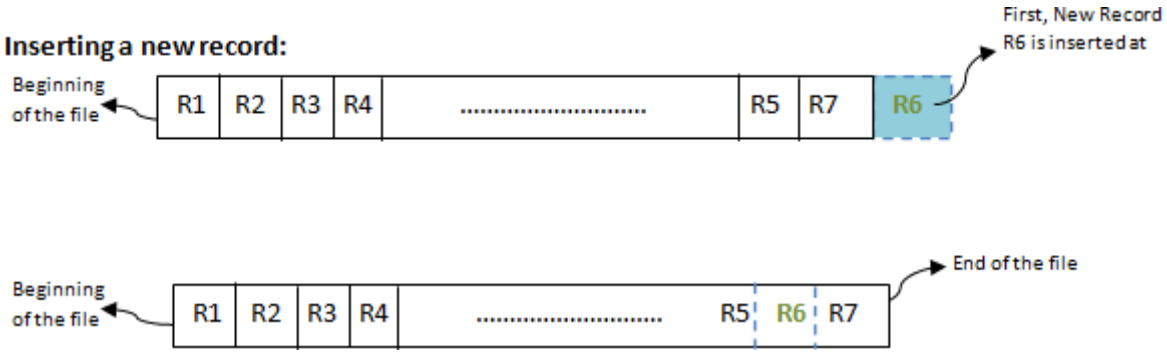
In the diagram above, R1, R2, R3 etc are the records. They contain all the attribute of a row. i.e.; when we say student record, it will have his id, name, address, course, DOB etc. Similarly R1, R2, R3 etc can be considered as one full set of attributes.



In the second method, records are sorted (either ascending or descending) each time they are inserted into the system. This method is called **sorted file method**. Sorting of records may bebased on the primary key or on any other columns. Whenever a new record is inserted, it will be inserted at the end of the file and then it will sort – ascending or descending based on key value and placed at the correct position. In the case of update, it will update the record and then sort thefile to place the updated record in the right place. Same is the case with delete.



Inserting a new record:



Advantages:

- Simple to understand.
- Easy to maintain and organize
- Loading a record requires only the record key.
- Relatively inexpensive I/O media and devices can be used.
- Easy to reconstruct the files.
- The proportion of file records to be processed is high.

Disadvantages:

- Entire file must be processed, to get specific information.
- Very low activity rate stored.
- Transactions must be stored and placed in sequence prior to processing.
- Data redundancy is high, as same data can be stored at different places with differentkeys.

- Impossible to handle random enquiries.

Conclusion: This program gives us the knowledge sequential file organization..

Program:

```
#include <iomanip.h>
#include <iostream.h>
#include <fstream.h>
#include <conio.h>
#include <string.h>
struct student
{int rollno;
char name[20];
float marks;
int status;
};
class sequential
{ char master1[30];
fstream mas;
public:
sequential(char *a)
{
strcpy(master1,a);
mas.open(master1,ios::binary|ios::in);
if(mas.fail())
mas.open(master1,ios::binary | ios::out);
mas.close();
}
void read(); //dispaly master file void insert(student rec1);
int Delete(int rollno);
int search(int rollno);
void pack();
void update();
void display(int recno) // display a particular record
{ student rec1;
mas.open(master1,ios::binary | ios::in | ios::nocreate);
mas.seekg(recno*sizeof(student),ios::beg);
mas.read((char*) &rec1,sizeof(student));
cout<<"\n"<<rec1.rollno<<""<<rec1.name<<""<<setprecision(2)<<rec1.marks;
mas.close();
}
};
void main()
{ sequential object("master.txt");
```

```
int rollno,op,recno;
student rec1;
do
{ cout<<"\n\n1)Read(Print)\n2)Insert\n3)Delete\n4)Update";
cout<<"\n5)Search\n6)Pack\n7)Quit";
cout<<"\nEnter Your Choice:";
cin>>op;
switch(op)
{ case 1: object.read();break; case 2: cout<<"\nEnter a record to be inserted(roll
no,name,marks : ";
cin>>rec1.rollno>>rec1.name>>rec1.marks;
object.insert(rec1);
break;
case 3: cout<<"\nEnter the roll no.:";
cin>>rollno;
object.Delete(rollno);
break;
case 4: object.update(); break;
case 5: cout<<"\nEnter a roll no. : ";
cin>>rollno;
recno=object.search(rollno);
if(recno>=0)
{ cout<<"\n Record No.: "<< recno;
object.display(recno);
}
else
cout<<"\nRecord Not Found ";
break;
case 6: object.pack();break;
}
}while(op!=7);
}
void sequential::read()
{ student crec; int i=1,n;
cout<<"\n*********Data File*********\n";
mas.open(master1,ios::binary | ios::in | ios::nocreate);
mas.seekg(0,ios::end);/*go to the end of file */
n=mas.tellg()/sizeof(student);
mas.seekg(0,ios::beg);
for(i=1;i<=n;i++)
{ mas.read((char*)&crec,sizeof(student));
if(crec.status==0)
cout<<"\n"<<i<<") "<<crec.rollno<<""<<crec.name<<""<<setprecision(2)<<crec.marks;
else
cout<<"\n"<<i<<") "<<" ****** deleted *********";
}
mas.close();
}
void sequential::insert(student rec1)
```

```
{ student crec;
int n,i,k;
mas.open(master1,ios::in | ios::out | ios::nocreate);
rec1.status=0;
mas.seekg(0,ios::end);/*go to the end of file */
n=mas.tellg()/sizeof(student);
if(n==0)
{
mas.write((char*)&rec1,sizeof(student)); mas.close();
return;
}
/* Shift records until the point of insertion */
i=nÂ1;
while(i>=0)
{ mas.seekg(i*sizeof(student),ios::beg);
mas.read((char*)&crec,sizeof(student));
if(crec.rollno>rec1.rollno)
{ mas.seekp((i+1)*sizeof(student),ios::beg);
mas.write((char*)&crec,sizeof(student));
}
else
break;
iÂÂ;
}
/*insert the record at (i+1)th position */
i++;
mas.seekp(i*sizeof(student),ios::beg);
mas.write((char*)&rec1,sizeof(student));
mas.close();
}
int sequential::Delete(int rollno)
{ student crec;
int i,n; mas.open(master1,ios::in | ios::out | ios::nocreate);
mas.seekg(0,ios::end);/*go to the end of file */
n=mas.tellg()/sizeof(student);
mas.seekg(0,ios::beg);
for(i=0;i<n;i++)
{ mas.read((char*)&crec,sizeof(student));
if(crec.status==0)
{
if(crec.rollno>rollno)
{cout<<"\nRecord does not exist …";
mas.close();
return(0);
}
if(crec.rollno==rollno)
{crec.status=1;
mas.seekp(i*sizeof(student),ios::beg);
mas.write((char*)&crec,sizeof(student));
```

```
mas.close();
return(1);
}
}
}
return(0);
}
int sequential::search(int rollno){ student crec;
int i,n;
mas.open(master1,ios::in | ios::out | ios::nocreate);
mas.seekg(0,ios::end);/*go to the end of file */
n=mas.tellg()/sizeof(student);
mas.seekg(0,ios::beg);
for(i=0;i<n;i++)
{ mas.read((char*)&crec,sizeof(student));
if(crec.status==0)
{
if(crec.rollno>rollno)
{mas.close();
return(Â1);
}
if(crec.rollno==rollno)
{ mas.close();
return(i);
}
}
}
return(Â1);
}
void sequential::pack()
{ fstream temp;
student crec;int i,n;
mas.open(master1,ios::binary | ios::in);
temp.open("temp.txt",ios::out | ios::trunc | ios::binary);
mas.seekg(0,ios::end);/*go to the end of file */
n=mas.tellg()/sizeof(student);
mas.seekg(0,ios::beg);
for(i=0;i<n;i++)
{ mas.read((char*)&crec,sizeof(student));
if(crec.status==0)
temp.write((char*)&crec,sizeof(student));
}
mas.close();
temp.close();
temp.open("temp.txt",ios::binary | ios::in);
mas.open(master1,ios::binary | ios::out | ios::trunc);
temp.seekg(0,ios::end);/*go to the end of file */
n=temp.tellg()/sizeof(student);
temp.seekg(0,ios::beg);
```

```
for(i=0;i<n;i++)
{ temp.read((char*)&crec,sizeof(student));
mas.write((char*)&crec,sizeof(student));
}
mas.close();
temp.close();
}void sequential::update()
{ int rollno;
student rec1;
cout<<"\n Enter the rollno of the record to be updated : ";
cin>>rollno;
cout<<"\nEnter a new record(roll no. name marks : ";
cin>>rec1.rollno>>rec1.name>>rec1.marks;
if(Delete(rollno))
insert(rec1);
else
cout<<"\n Record not found :";
}
```