UNIT V FILE PROCESSING 9

Files – Types of file processing: Sequential access, Random access – Sequential access file - Random access file - Command line arguments.

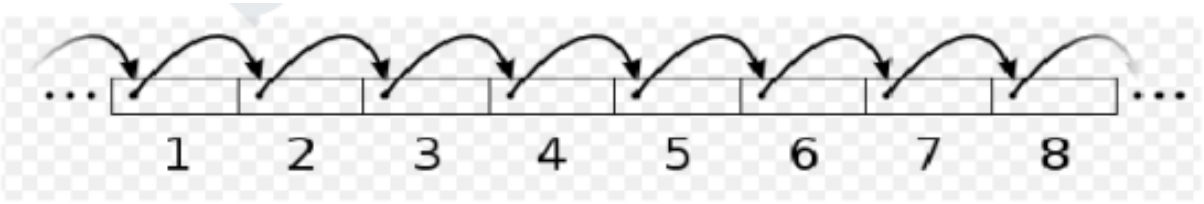
TYPES OF FILE PROCESSING: SEQUENTIAL ACCESS, RANDOM ACCESS

In computer programming, the two main types of file handling are:

- Sequential access
- Random access

Sequential access

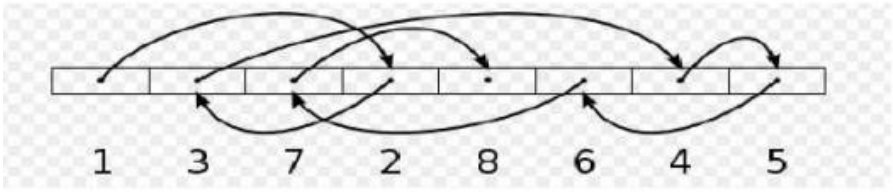
In this type of files data is kept in sequential order if we want to read the last record of the file, we need to read all records before that record so it takes more time.



Sequential access to file

Random access

In this type of files data can be read and modified randomly .If we want to read the last record we can read it directly. It takes less time when compared to a sequential file.



Random Access To File

There is no need to read each record sequentially, if we want to access a particular record. C supports these functions for random access file processing.

1. fseek()
2. ftell()
3. rewind()

fseek():

It is used to move the reading control to different positions using fseek function.

Syntax:

fseek(file pointer, displacement, pointer position);

Where

file pointer ---- It is the pointer which points to the file.

displacement ---- It is positive or negative. This is the number of bytes which are skipped backward (if negative) or forward(if positive) from the current position. This is attached with L because this is a long integer.

Pointer position:

This sets the pointer position in the file. The three values are,

- 0 - Beginning of file
- 1 - Current position
- 2 - End of file

Example:

1) fseek(p,10L,0)

0 means pointer position is at the beginning of the file,from this statement pointer position is skipped 10 bytes from the beginning of the file.

2)fseek(p,5L,1)

1 means current position of the pointer position. From this statement pointer position is skipped 5 bytes forward from the current position.

3)fseek(p,-5L,1)

From this statement pointer position is skipped 5 bytes backward from the current position.

ftell()

It tells the byte location of the current position of the cursor in file pointer.

rewind()

It moves the control to the beginning of the file.

Example program for fseek():

Write a program to read last ‘n’ characters of the file using appropriate file functions(Here we need fseek() and fgetc())

```
#include<stdio.h>
#include<conio.h>
void main()
{
```

```
        FILE *fp;
        char ch;
        clrscr();
        fp=fopen("file1.c", "r");
        if(fp==NULL)
                printf("file cannot be opened");
        else
        {
                printf("Enter value of n to read last ‘n’ characters");
                scanf("%d",&n);
                fseek(fp,-n,2);
                while((ch=fgetc(fp))!=EOF)
                {
                        printf("%c\t",ch);}
                }
        }
        fclose(fp);
        getch();
}
```

COMMAND LINE ARGUMENTS

Command line argument is a parameter supplied to the program when it is invoked. Command line argument is an important concept in C programming. It is mostly used when you need to control your program from outside. Command line arguments are passed to the main() method.

Syntax:

int main(int argc, char *argv[])

Here argc counts the number of arguments on the command line and argv[ ] is a pointer array which holds pointers of type char which points to the arguments passed to the program.

Example:

```
#include <stdio.h>
#include <conio.h>
int main(int argc, char *argv[])
```

```
{
        int i;
        if( argc >= 2 )
        {
                printf("The arguments supplied are:\n");
                for(i = 1; i < argc; i++)
                {
                        printf("%s\t", argv[i]);
                }
        }
        else
        {
                printf("argument list is empty.\n");
        }
        return 0;
}
```

Remember that argv[0] holds the name of the program and argv[1] points to the first command line argument and argv[n] gives the last argument. If no argument is supplied, argc will be 1.
