Paper 102: Programming & Problem solving through C

Lecture-21:Unit-III
Files-II

Aiusha V.Hujon, Dept. of Computer Science, SAC

Binary mode

- Files can also be operated using binary mode
- There are three differences between text mode and binary mode
  - Handling of newlines
  - Representation of end of file
  - Storage of numbers

Handling of newlines

- In text mode a new line character is converted into carriage return-linefeed combination before being written to the disk
  - "\r\n"
- Similarly the carriage return-linefeed combination on the disk is converted back into a newline when the file is read by a C program
- In binary mode these conversion does not takes place
- The opening modes in binary are the same modes as in text modes except that each opening mode is followed by a letter 'b'
  - E.g,' rb' or 'wb' instead of the text mode 'r' or 'rt'

Representation of end of file

- In text mode a special character whose ASCII value is '26' is inserted after the last character in the file to mark the end of a file
  - If this character is detected by any read function it returns EOF
- There is no such special character in Binary mode
  - It keep track of the end of file by the numbers of characters present in the directory entry of the file
- The two modes are not compatible
  - Problems like the number '26' if stored in the file in binary mode, when open in text mode this might be treated as end of file marker

Storage of numbers

- The only function that allows storage of numbers is fprintf()
- This function stores character as character, i.e., one byte.
  - Numbers however are stored as strings of characters
  - Hence 141 is stored character by character i.e., three bytes and 123.45 is stored as 6 byte string
- Thus numbers with more digits would require more disk space
- In binary mode when files are operated using the two functions fread() and fwrite() would store numbers in the same way as they are stored in memory

fwrite and fread

- fwrite appends a specified number of equal-sized data items to an output file
- Declaration:
  - size_t fwrite(const void *ptr, size_t size, size_t n, FILE*stream);

ptr Pointer to any object; the data written begins at ptr

size- Length of each item of data

n Number of data items to be appended

stream Specifies output file

  - The total number of bytes written is (n * size)
- Return Value:
  - On success, returns the number of items (not bytes) actually written.
  - On error, returns a short count.
  - Reads data from a stream

fwrite and fread

- fread reads a specified number of equal-sized data items from an input stream into a block.
- Declaration:
  - size_t fread(void *ptr, size_t size, size_t n, FILE *stream);

ptr	Points to a block into which data is read
size	Length of each item read, in bytes
n	Number of items read
stream	Points to input stream

The total number of bytes read is (n * size).

E.g.: fwrite (&e, sizeof (e), 1, fp) ;

fread (&e, sizeof (e), 1, fp)

- Return Value:
  - On success, fread returns the number of items (not bytes) actually read
  - On end-of-file or error, fread returns a short count (possibly 0)

fseek

- Declaration
  - int fseek(FILE *stream, long offset, int whence);
  - Stream — stream whose file pointer fseek sets
  - Offset — difference in bytes between whence and the new position for text mode stream. Offset should be 0 or a value returned by ftell()
  - Whence — one of the three SEEK_xxx fle pointer locations
  - SEEK_SET — 0 -is an offset measured relative to the beginning of the file
  - SEEK_CUR — 1- is an offset measured relative to the current position of the file
  - SEEK_END — 2- is an offset measured relative to the end of the file
- Return value
  - On success it returns 0
  - On failure it returns a non-zero value
- fseek(fp,recsize,SEEK_CUR);

Example of fwrite

```
/* Receives records from keyboard and writes them to a file in binary mode */
#include "stdio.h"
main( )
{
        FILE *fp ;
        char another = 'Y' ;
        struct emp
        {
        char name[40] ;
        int age ;
        float bs ;
        } ;
        struct emp e ;
        fp = fopen ( "EMP.DAT", "wb" ) ;
        if ( fp == NULL )
        {
                puts ( "Cannot open file" ) ;
                exit( ) ;
        }
```

Example of fwrite – cont…

```
while ( another == 'Y' )
{
        printf ( "\nEnter name, age and basic salary: " ) ;
        scanf ( "%s %d %f", e.name, &e.age, &e.bs ) ;
        fwrite ( &e, sizeof ( e ), 1, fp ) ;
        printf ( "Add another record (Y/N) " ) ;
        fflush ( stdin ) ;
        another = getche( ) ;
}
fclose ( fp ) ;
}
```

And here is the output…

Enter name, age and basic salary: Suresh 24 1250.50

Add another record (Y/N)Y

Enter name, age and basic salary: Ranjan 21 1300.60

Add another record (Y/N)Y

Enter name, age and basic salary: Harish 28 1400.70

Add another record (Y/N) N

Example of fread

```
/* Reads records from binary file and displays them on VDU */
#include "stdio.h"
main( )
{
      FILE *fp ;
      struct emp
      {
          char name[40] ;
          int age ;
          float bs ;
      } ;
      struct emp e ;
      fp = fopen ( "EMP.DAT", "rb" ) ;
      if ( fp == NULL )
      {
          puts ( "Cannot open file" ) ;
          exit( ) ;
      }
      while ( fread ( &e, sizeof ( e ), 1, fp ) == 1 )
          printf ( "\n%s %d %f", e.name, e.age, e.bs ) ;
      fclose ( fp ) ;
}
```