File Management

File management is what you have, and how you want to manipulate it. - Anonymous

Why File Management?

- In real life, we want to store data permanently so that later we can retrieve it and reuse it.
- A file is a collection of characters stored on a secondary storage device like hard disk, or pen drive.
- There are two kinds of files that programmer deals with:
  - Text Files are human readable and it is a stream of plain English characters
  - Binary Files are computer readable, and it is a stream of processed characters and ASCII symbols

Text File

Hello, this is a text file. Whatever written here can be read easily without the help of a computer.

Binary File

110100110101000101101110101
011101011101001101010001011
011101010111010111010011

File Opening Modes

- We can perform different operations on a file based on the file opening modes

Mode	Description
r	Open the file for reading only. If it exists, then the file is opened with the current contents; otherwise an error occurs.
w	Open the file for writing only. A file with specified name is created if the file does not exists. The contents are deleted, if the file already exists.
a	Open the file for appending (or adding data at the end of file) data to it. The file is opened with the current contents safe. A file with the specified name is created if the file does not exists.
r+	The existing file is opened to the beginning for both reading and writing.
w+	Same as w except both for reading and writing.
a+	Same as a except both for reading and writing.

Note: The main difference is w+ truncate the file to zero length if it exists or create a new file if it doesn't. While r+ neither deletes the content nor create a new file if it doesn't exist.

File Handling Functions

- Basic file operation performed on a file are opening, reading, writing, and closing a file.

Syntax	Description
`fp=fopen(file_name, mode);`	This statement opens the file and assigns an identifier to the FILE type pointer fp. Example: `fp = fopen("printfile.c","r");`
`fclose(filepointer);`	Closes a file and release the pointer. Example: `fclose(fp);`
`fprintf(fp, “control string”, list);`	Here fp is a file pointer associated with a file. The control string contains items to be printed. The list may includes variables, constants and strings. Example: `fprintf(fp, "%s %d %c", name, age, gender);`

File Handling Functions

Syntax	Description
`fscanf(fp, “control string”, list);`	Here `fp` is a file pointer associated with a file. The control string contains items to be printed. The list may includes variables, constants and strings. Example: `fscanf(fp, "%s %d", &item, &qty);`
`int getc( FILE *fp);`	`getc()` returns the next character from a file referred by `fp`; it require the `FILE` pointer to tell from which file. It returns `EOF` for end of file or error. Example: `c = getc(fp);`
`int putc(int c, FILE *fp);`	`putc()` writes or appends the character `c` to the `FILE` `fp`. If a `putc` function is successful, it returns the character written, `EOF` if an error occurs. Example: `putc(c, fp);`

File Handling Functions

Syntax	Description
`int getw(FILE *pvar);`	`getw()` reads an integer value from `FILE` pointer `fp` and returns an `integer`. Example: `i = getw(fp);`
`putw(int, FILE *fp);`	putw writes an integer value read from terminal and are written to the `FILE` using `fp`. Example: `putw(i, fp);`
`EOF`	`EOF` stands for “End of File”. `EOF` is an integer defined in `<stdio.h>` Example: `while(ch != EOF)`

Write a C program to display content of a given file.

Program

```c
1  #include <stdio.h>
2  void main()
3  {
4      FILE *fp; //p is a FILE type pointer
5      char ch; //ch is used to store single character
6      fp = fopen("file1.c","r"); //open file in read mode and store file pointer in p
7      do {  //repeat step 9 and 10 until EOF is reached
8          ch = getc(fp); //get character pointed by p into ch
9          putchar(ch); //print ch value on monitor
10     }while(ch != EOF); //condition to check EOF is reached or not
11     fclose(fp); //free up the file pointer pointed by fp
12 }
13
```

Write a C program to copy a given file.

Program

```c
#include <stdio.h>
void main()
{
    FILE *fp1, *fp2; //p and q is a FILE type pointer
    char ch; //ch is used to store temporary data
    fp1 = fopen("file1.c","r"); //open file “file1.c” in read mode
    fp2 = fopen("file2.c","w"); //open file “file2.c” in write mode
    do { //repeat step 9 and 10 until EOF is reached
        ch = getc(fp1); //get character pointed by p into ch
        putc(ch, fp2); //print ch value into file, pointed by pointer q
    }while(ch != EOF); //condition to check EOF is reached or not
    fclose(fp1); //free up the file pointer p
    fclose(fp2); //free up the file pointer q
    printf("File copied successfully...");
}
```

File Positioning Functions

- `fseek`, `ftell`, and `rewind` functions will set the file pointer to new location.
- A subsequent read or write will access data from the new position.

Syntax	Description
`fseek(FILE *fp, long offset, int position);`	`fseek()` function is used to move the file position to a desired location within the file. **`fp`** is a FILE pointer, **`offset`** is a value of datatype `long`, and **`position`** is an `integer` number. **Example**: /* Go to the end of the file, past the last character of the file */ `fseek(fp, 0L, 2);`
`long ftell(FILE *fp);`	`ftell` takes a file pointer and returns a number of datatype `long`, that corresponds to the current position. This function is useful in saving the current position of a file. **Example**: /* n would give the relative offset of the current position. */ `n = ftell(fp);`

File Positioning Functions

Syntax	Description
`rewind(fp);`	`rewind()` takes a file pointer and resets the position to the start of the file. **Example**: /* The statement would assign 0 to n because the file position has been set to the start of the file by rewind. */ `rewind(fp);`

Write a C program to count lines, words, tabs, and characters

Program

```c
#include <stdio.h>
void main()
{
    FILE *p;
    char ch;
    int ln=0,t=0,w=0,c=0;
    p = fopen("text1.txt","r");
    ch = getc(p);
    while (ch != EOF) {
        if (ch == '\n')
            ln++;
        else if(ch == '\t')
            t++;
        else if(ch == ' ')
            w++;
        else
```

Program (contd.)

```c
            c++;

        ch = getc(p);
    }
    fclose(p);
    printf("Lines = %d, tabs = %d, words = %d, characters = %d\n",ln, t, w, c);
}
```

Output

Lines = 22, tabs = 0, words = 152, characters = 283

Practice Programs

1) Write a C program to write a string in file.
2) A file named data contains series of integer numbers. Write a C program to read all numbers from file and then write all the odd numbers into file named “odd” and write all even numbers into file named “even”. Display all the contents of these file on screen.
3) Write a C program to read name and marks of n number of students and store them in a file.
4) Write a C program to print contents in reverse order of a file.
5) Write a C program to compare contents of two files.
6) Write a C program to copy number of bytes from a specific offset to another file.
7) Write a C program to convert all characters in UPPER CASE of a File.

Thank you