Introduction To File Management & Its Functions

Computer Programming & Utilization

Introduction

- The functions that we use such as scanf( ) & printf( ) are there to read and write data. These I/O functions always use the terminal (Keyboard and Screen) as the target place.

- These functions work where the data is small, but the entire process becomes cumbersome and time consuming when it comes to many real-life problems which involve large volumes of data through terminals & due to this the entire data is lost when either a program is terminated or computer is turned off.

- So a flexible approach was brought in, leading to the concept of files in which data can be stored on the disks and read whenever necessary, without destroying the data.

What is a File?

- A *file* is a place on the disk where a group of related data is stored.

- Computers store files to secondary storage so that the contents of files remain intact when a computer shuts down.

- When a computer reads a file, it copies the file from the storage device to memory; when it writes to a file, it transfers data from memory to the storage device.

- Two distinct ways to perform file operations in C:-

  low-level I/O – Uses UNIX system calls.

  high-level I/O – Uses functions in C's standard I/O library.

High Level I/O Functions

Function Name	Operation
fopen()	Creates a new file/Opens an existing file for use.
fclose()	Closes a file which has been opened for use.
getc()	Reads a character from a file.
putc()	Writes a character to a file.
fprintf()	Writes a set of data values to a file.
fscanf()	Reads a set of data values to a file.
getw()	Reads an integer from file.
putw()	Writes an integer from file.
fseek()	Sets the position to a desired a point in the file.
ftell()	Gives the correct position in the file.
rewind()	Sets the position to the beginning of the file.

Defining & Opening a File

- *Filename* is a string of characters that make up a valid filename for the operating system.
- *Data structure* of a file is defined as **FILE** in the library of standard I/O functions definitions.
- C uses a structure called FILE (defined in stdio.h) to store the attributes of a file. **FILE** is a defined data type.
- General format for defining & opening a file:-

```
FILE *fp;

fp = fopen("filename" , "mode");
```

- The first statement declares the variable **fp** as a “pointer to the data type **FILE**”.

- The second statement opens the file *filename* and assigns an identifier to the **FILE** type pointer **fp.** It also specifies the purpose of opening this file. The *mode* does this job.

- Modes:-

  r – Open the file for reading only.
  w – Open the file for writing only.
  a – Open the file for appending data to it.

```
FILE *p1, *p2;

p1 = fopen(“CPU” , “r”);
p2 = fopen(“Data” , “w”);
```

Closing a File

- Closing a file ensures that all outstanding information associated with the file is flushed out from the buffers and all links to the file are broken.
- Syntax for closing a file:-

```
fclose(file_pointer);
```

- Example:-

```
FILE *p1, *p2;
p1 = fopen(“cpu” , “r”);
p2 = fopen(“data” , “w”);
……………….
fclose(p1);
fclose(p2);
```

The *getc* and *putc* Functions

- *getc* and *putc* are the simplest file I/O functions. These are analogous to *getchar* and *putchar* functions and handle one character at a time.

- *getc* is used to read a character from a file that has been opened, while *putc* is used to write a character to the file that has been opened.

- <u>Syntax for *getc*</u>:-

```
identifier = getc(file pointer);
```

- <u>Syntax for *putc*</u>:-

```
putc(identifier , file pointer);
```

- Example:-

```
FILE *fp1, *fp2;

fp1= fopen("input.txt" , "r");
fp2 = fopen("output.txt" , "w");
char ch;
ch = getc (fp1);
putc(ch,fp2);
fclose(fp1);
fclose(fp2);
```

The *getw* and *putw* Functions

- The *getw* and *putw* are integer-oriented functions. They are similar to *getc* and *putc* functions and are used to read and write integer values.

- Syntax for *getw*:-

```
getw(file pointer);
```

- Syntax for *putw*:-

```
putw(integer , file pointer);
```

The *fprintf* and *fscanf* Functions

- Most compilers support functions *fprintf* and *fscanf* ,that can handle a group of mixed data simultaneously.

- The functions *fprintf* and *fscanf* are identical to the familiar *printf* and *scanf* functions, except of course that they work on files.

- Syntax & Example of *fprintf*:-

```
fprintf(fp ,“Control String”, list);

fprintf(fp,“%s %d %f”,name,age,8);
```

- Syntax & Example of *fscanf*:-

```
fscanf(fp ,“Control String”, list);

fprintf(fp,“%s %d”,name,&quantity);
```

Thank You