# Basic File Processing Operation
# Handling files in C Language

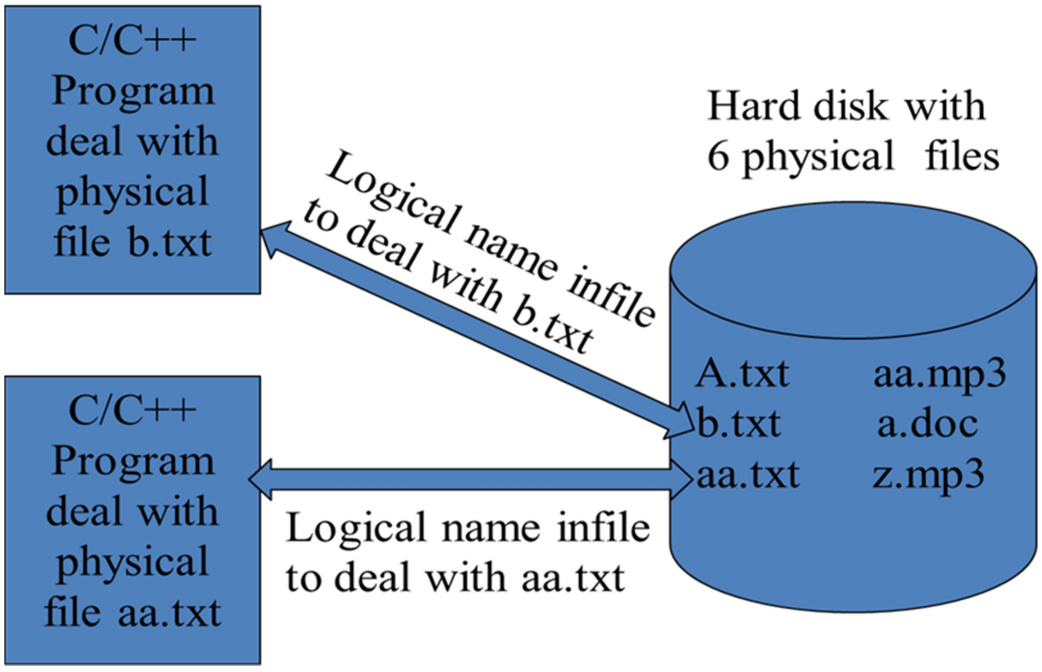
## Contents of Lecture:

- Physical File vs. Logical File
- Opening and Closing Files
- Reading and Writing
- Sample Program
- Full Methods

## Physical File vs. Logical File

- **What is a physical file?**
  - ✓ It is a collection of bytes stored on a disk
  - ✓ It has a physical name; e.g., A.txt
  - ✓ It is used once when opening the file

- **What is a logical file?**
  - ✓ It is a channel that connects the program to the corresponding physical file.
  - ✓ It has a logical name, which is a variable inside the program; e.g, infile, fp.
  - ✓ It is used as many times as needed when opening, reading, writing, appending, or closing the file.

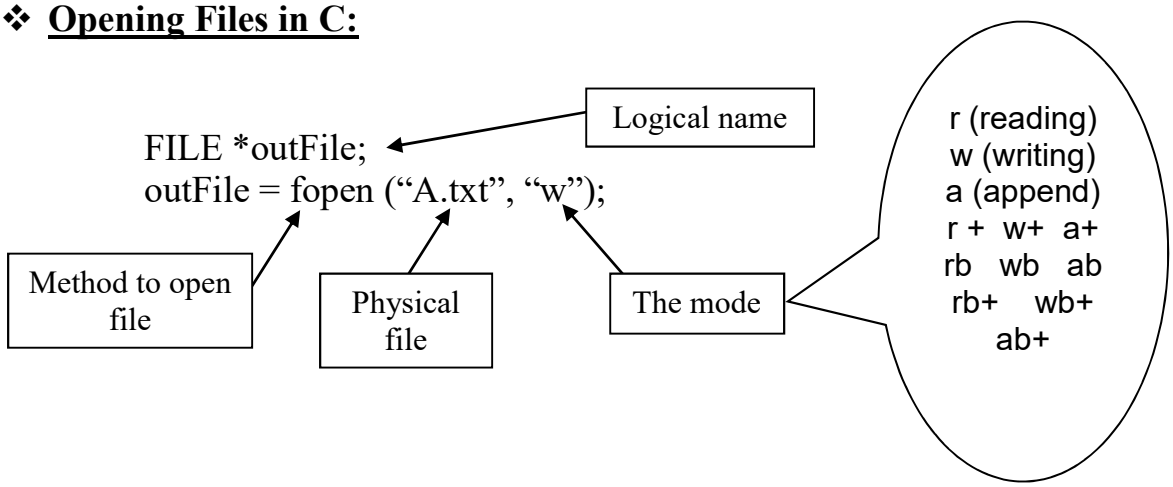


# Opening and Closing Files:

## ❖ Opening Files:

- Opening a file makes it ready for use. There are two options:
  - Open an existing file.
  - Create a new file.
- When opening a file, we are positioned at the beginning of the file.

## ❖ Opening Files in C:




## ❖ Closing Files:

- Makes the logical file name available for another physical file (it's like hanging up the telephone after a call).
  - After closing, the logical name may be used again with another physical file

- The bytes are not sent one by one to the physical file. Instead, they are stored in a buffer and sent as a block. When the file is closed, the leftover from the buffer is flushed to the file.

- If the files is not closed in the program, the operating system closes it at the end of the program execution. However, if the program terminates abnormally, data may be lost.

- In C: fclose(outFile);

# Reading and Writing:

## ❖ Reading:

- Read(Source_file, Destination_addr, Size)
  - Source_file = location the program reads from, i.e., its logical file name
  - Destination_addr = first address of the memory block where we want to store the data.
  - Size = how much information is being brought in from the file (byte count).

- Methods used to read from files:

| Method | shortcut | Usage |
|---|---|---|
| fgetc | File Get Character | Read one Character from file |
| fgets | File Get String | Read string of Character from file |
| fscanf | File Scan Format | Read formatted text from file |
| fread | File Read | Read record from file |

- Methods syntax:

| Method | Syntax | Example |
|---|---|---|
| fgetc | fgetc(file_pointer); | ch = fgetc(fp); |
| fgets | fgets(string_var , No_Of_Character , file_pointer); | fgets( string ,80 , fp ); |
| fscanf | fscanf( file_pointer , " conversion specifires " , var(s)_name); | result = fscanf(my_file , %c %d %x" ,ch ,num , hex); |
| fread | fread(&block_var , block_size , No_of_block , file_pointer ); | fread(&emp,sizeof(emp),1,my_file); |

## ❖ Writing:

- Write(Destination_file, Source_addr, Size)
  - Destination_file = the logical file name where the data will be written.
  - Source_addr = first address of the memory block where the data to be written is stored.
  - Size = the number of bytes to be written.

- Methods used to write to files:

| Method | shortcut | Usage |
|---|---|---|
| fputc | File Put Character | write one Character to file |
| fputs | File Put String | Write string of Character to file |
| fprintf | File Print Format | Write formatted text to file |
| fwrite | File Write | Write record to file |

- Methods syntax:

| Method | Syntax | Example |
|---|---|---|
| fputc | fputc ( var_name , file_pointer ); | fputc(ch,fp); |
| fputs | fputs ( string_var , file_pointer); | fputs("this is a text file",pf); |
| fprintf | fprintf( file_pointer , " text and \| or conversion specifires " , var(s)_name); | fprintf(my_file , %c \t%d \t %x \n " ,ch ,ch ,ch ); |
| fwrite | fwrite(&block_var , block_size , No_of_block , file_pointer ); | fwrite(&emp , sizeof(emp) ,1 , my_file); |

## Sample Program:

A program that displays the contents of a file on the screen

**Algorithm:**

- Open the file for input
- While there are characters to read:
  - Read a character from the file
  - Write the character to the screen
- Close the file

### Sample Program in C:

```
struct employee
        {
          char name[20];
          int sal;
        };

void main()
{
 FILE *my_file;
 struct employee emp;
 my_file=fopen("c:\\emp.txt ","rb+");
```

```
if(my_file = = NULL)
  {
    printf("Error in opening file");
    exit(1);
   }

while(! feof(my_file))
     {
       fread(&emp,sizeof(emp),1,my_file);
       printf("The Emp Name :%s \t ", emp.name);
       printf("The Emp Sal : %f\n", emp.sal ");
     }

fclose(my_file);
 getch();
 }
```

## Full Methods:

1. **Write a method writeC() ask user to enter one character from keyboard, then write the character to the physical file alphabet.txt.**

```
void writeC()
{
  FILE *fp;
  fp = fopen("c:\\alphabet.txt","w");
  if(fp = = NULL)
  {
     Printf("Error in opening file");
     Exit(1);
   }

 char ch;
 ch = getche();

 fputc(ch,fp);

 fclose(fp);
}
```

**2. Write a method readC() which read one character from the physical file alphabet.txt then print the character to screen.**

```
void readC()
{
  FILE *fp;
  fp = fopen("c:\\alphabet.txt","r");
  if(fp = = NULL)
  {
     printf("Error in opening file");
     exit(1);
   }
  char ch;

  ch = fgetc(fp);

  putchar(ch);
  fclose(fp);
}
```

**3. Write a method writeS() ask user to enter string of character from keyboard until press enter, then write the string to the physical file std.txt.**

```
void writeS()
{
  FILE *my_file;
  my_file=fopen("c:\\std.txt","w");

 if(my_file = = NULL)
  {
     printf("Error in opening file");
     exit(1);
   }

 char string[80];
    while (1)
    {
     gets(string);
     if(strlen(string ) = =  0)
          break;
    fputs(string,my_file);
     }
  fclose(my_file);
}
```

4. **Write a method readS() which read string of character from the physical file std.txt then print the character to screen.**

```
void read()
{
  FILE *my_file;
  int result;
  my_file=fopen("c:\\std.txt","r");

 if(my_file = = NULL)
   {
      printf("Error in opening file");
      exit(1);
    }
  char string[80];

 while(1)
  {
     fgets(string , 80 , my_file) ;
     result = feof(my_file);
     if(result !=0 )
        break;
     puts(string);
   }
 fclose(my_file);
 }
```

5. **Write a method writeASCII() which write part of the ASCII table to the physical file ascii.txt using fprintf.**

```
void writeASCII()
{
  FILE *my_file;
  char ch ;
  my_file=fopen("ascii.txt","w");
  if(my_file = = NULL)
  {
     printf("Error in opening file");
     exit(1);
   }

 fprintf(my_file ,"Letter\t Decimal \t Hexadecimal \n");
 for(ch = 32 ; ch<=127 ; ch++ )
     fprintf(my_file , %c  \t%d  \t %x \n " ,ch ,ch ,ch );
  fclose(my_file);
}
```

**6. Write a method readASCII () which read data from the physical file ascii.txt then print the character to screen.**

```
void read()
{
  FILE *my_file;
  char string[80] ;
  int ch , num , hex;
  int result;
  my_file=fopen("ascii.txt","r");
  if(my_file = = NULL)
  {
     printf("Error in opening file");
     exit(1);
   }

fgets(string , 80 , my_file);
puts(string);
while (1)
{
  result = fscanf(my_file , %c %d %x" ,ch ,num , hex);
  if(result = = EOF)
        break;
  printf("%c  \t%d  \t %x \n "  , ch ,num , hex);
}

fclose(my_file);
}
```