CSE102
Computer Programming

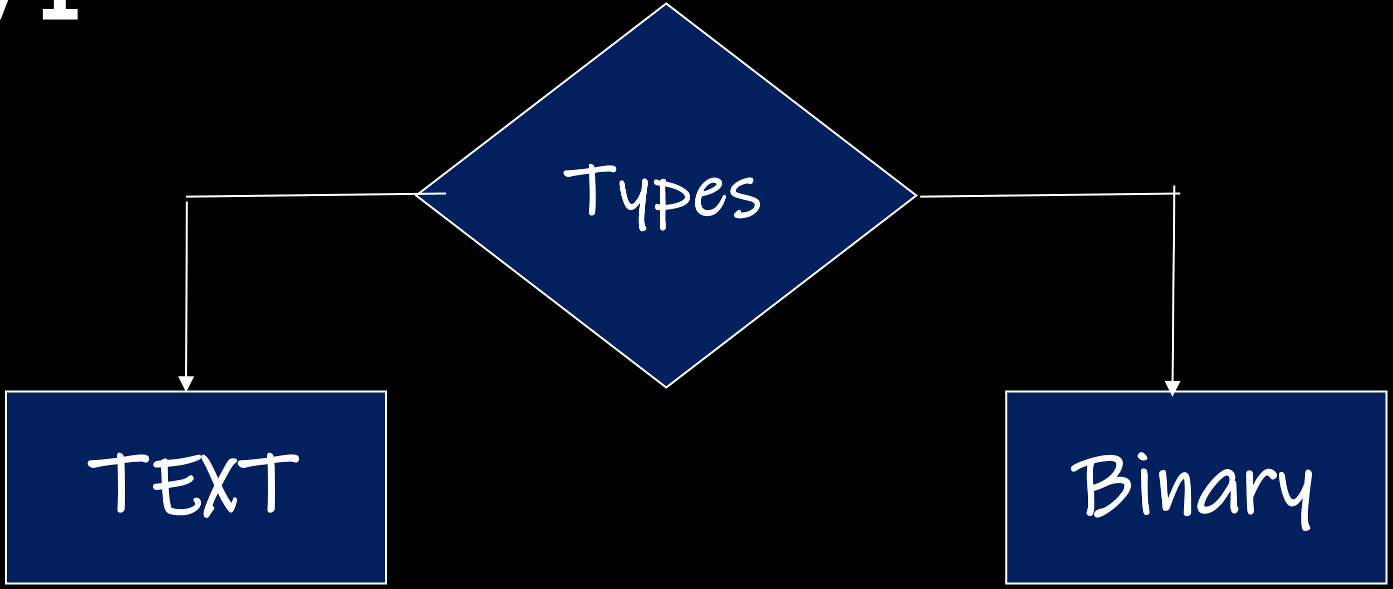
Copyright 2004 by Randy Glasbergen.
www.glasbergen.com
MEMO
GLASBERGEN
"I am not disorganized — I know *exactly* where everything is! The newer stuff is on top and the older stuff is on the bottom."

Credits: programiz

Background

- A collection of related data that a computers treats as a single unit.
- 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

File Types



- The normal .txt files
- Created using Notepad or any simple text editors
- Take minimum effort to maintain
- Are easily readable
- Provide least security and takes bigger storage space.

- Mostly the .bin files
- Data stored in the binary form (0's and 1's).
- Can hold higher amount of data
- Are not readable easily
- Provides a better security than text files..

Syntax

$$\underbrace{\texttt{FILE}}_{\text{Keyword}}\ \overbrace{\texttt{*fp}}^{\text{Variable}}\ ;$$

- Note that the variable used is of type **POINTER.**
- This declaration is needed for communication between the file and program.

File Operations

- Creating a new file
- Opening an existing file
- Closing a file
- Reading from and writing information to a file

Opening a file - for creation and edit

- Opening a file is performed using the library function in the **"stdio.h"** header file: **fopen()**.
-

$$\overbrace{\text{ptr}}^{\text{File pointer}} = \underbrace{\text{fopen}}_{\text{Keyword}}(\overbrace{\text{"fileopen"}}^{\text{FileName with full path}},\text{"mode"}) \xleftarrow{\text{refer next slide}}$$

- **Note:** While it is not mandatory to include the file extension in fileName, it is recommended that you follow the same when programming

File Modes

File Mode	Meaning of Mode	During Inexistence of file(I.e. If the file does not exist)
r	Open for reading.	If the file does not exist, fopen() returns NULL.
w	Open for writing.	If the file exists, its contents are overwritten. If the file does not exist, it will be created.
a	Open for append. i.e, Data is added to end of file.	If the file does not exists, it will be created.
r+	Open for both reading and writing.	If the file does not exist, fopen() returns NULL.
w+	Open for both reading and writing.	If the file exists, its contents are overwritten. If the file does not exist, it will be created.
a+	Open for both reading and appending.	If the file does not exists, it will be created.

Example – File Creation

```
FILE *fp;   /* file pointer*/
    char fName[20];
    printf("\nEnter file name to create :");
    scanf("%s",fName);

   /*creating (open) a file*/
       fp=fopen(fName,"w");
```

Reading and Writing into a File

- Reading a File
- Syntax:

```
fp=fopen(fName,"r");
printf("%c",getc(fp));
```

- Writing into a File
- Syntax

```
fp=fopen(fName,"w");
putc('A',fp);
```

Does this mean we can use only getc() and putc()? Explore fprintf () and fscanf() as well.

Closing a file

- The file (both text and binary) should be closed after reading/writing.

```
fclose(fp);
```

- Here fp is the file pointer associated with file to be closed.

Example: Write to a text file using fprintf()

```
#include <stdio.h>
#include <stdlib.h>

int main()
{
   int num;
   FILE *fptr;
   fptr = fopen("C:\\program.txt","w");

   if(fptr == NULL)
   {
      printf("Error!");
}
```

```
   printf("Enter num: ");
   scanf("%d",&num);

   fprintf(fptr,"%d",num);
   fclose(fptr);

   return 0;
}
```

Example 2: Read from a text file using fscanf()

```
#include <stdio.h>
#include <stdlib.h>
int main()
{
   int num;
   FILE *fptr;
   if ((fptr = fopen("C:\\program.txt","r")) == NULL)
    {
      printf("Error! opening file");
    }
    fscanf(fptr,"%d", &num);

   printf("Value of n=%d", num);

    fclose(fptr);

  return 0;
}
```

What about Binary Files?

What's the difference?

- Syntax for opening and closing a file are the same, the only thing that changes is the file mode.

File Mode	Meaning of Mode	During Inexistence of file(I.e. If the file does not exist)
rb	Open for reading in binary mode.	If the file does not exist, fopen() returns NULL.
wb	Open for writing in binary mode.	If the file exists, its contents are overwritten. If the file does not exist, it will be created.
ab	Open for append in binary mode. i.e, Data is added to end of file.	If the file does not exists, it will be created.
rb+	Open for both reading and writing in binary mode.	If the file does not exist, fopen() returns NULL.
wb+	Open for both reading and writing in binary mode.	If the file exists, its contents are overwritten. If the file does not exist, it will be created.
ab+	Open for both reading and appending in binary mode.	If the file does not exists, it will be created.

Reading and Writing into a Binary File

- Reading from a Binary File
    - Use the **fread()** function which takes 4 arguments
- Syntax:

```
fread(address_data,size_data,numbers_data,pointer_to_file);
```

- Writing into a Binary File
    - use the function **fwrite()** which takes 4 arguments.
- Syntax

```
fwrite(address_data,size_data,numbers_data,pointer_to_file);
```

Example 3: Write to a binary file using fwrite()

```
#include <stdio.h>
#include <stdlib.h>

struct threeNum
{
   int n1, n2, n3;
};
int main()
{
   int n;
   struct threeNum num;
   FILE *fptr;
if ((fptr = fopen("C:\\program.bin","wb")) == NULL){
       printf("Error! opening file");
   }
   for(n = 1; n < 5; ++n)
   {
      num.n1 = n;
      num.n2 = 5*n;
      num.n3 = 5*n + 1;
      fwrite(&num, sizeof(struct threeNum), 1, fptr);
   }
   fclose(fptr);
return 0;
}
```

Example 4: Read from a binary file using fread()

```
#include <stdio.h>
#include <stdlib.h>
struct threeNum
{
   int n1, n2, n3;
};
int main()
{
   int n;
   struct threeNum num;
   FILE *fptr;
if ((fptr = fopen("C:\\program.bin","rb")) == NULL){
       printf("Error! opening file");
   }
   for(n = 1; n < 5; ++n)
   {
      fread(&num, sizeof(struct threeNum), 1, fptr);
      printf("n1: %d\tn2: %d\tn3: %d", num.n1, num.n2,
num.n3);
   }
   fclose(fptr);
   return 0;
}
```

But what if you need a particular record in a file?

- Need to loop through all the records before it to get the record.
  - This will waste a lot of memory and operation time.
- Simpler way: use `fseek()`
- Syntax:

$$\texttt{fseek(FILE}\overbrace{\texttt{* stream,}}^{\text{File Pointer}}\ \texttt{long}\ \underbrace{\texttt{int offset,}}_{\text{Position of record to be found}}\ \overbrace{\texttt{int whence}}^{\text{the location where the offset starts}}\texttt{)}$$

Fseek() "whence" explained

Types of "Whence" in fseek

Whence	Meaning
SEEK_SET	Starts the offset from the beginning of the file.
SEEK_END	Starts the offset from the end of the file.
SEEK_CUR	Starts the offset from the current location of the cursor in the file.

Example 5: fseek()

```
#include <stdio.h>
struct threeNum
{
   int n1, n2, n3;
};
int main()
{
   int n;
   struct threeNum num; NULL.
   }
if((fptr = fopen("C:\\program.bin",
   "rb")) == NULL){
       printf("Error! opening
   file");}
```

```
  // Moves the cursor to the end of the file
  fseek(fptr, -sizeof(struct threeNum), SEEK_END);
for(n = 1; n < 5; ++n)
  {
     fread(&num, sizeof(struct threeNum), 1, fptr);
     printf("n1: %d\tn2: %d\tn3: %d\n", num.n1, num.n2,
num.n3);
     fseek(fptr, -2*sizeof(struct threeNum), SEEK_CUR);
  }
  fclose(fptr);
   return 0;
}
```

/*NOTE:This program will start reading the records from the file "program.bin" in the reverse order (last to first) and prints it.*/

A few Exercises – Try implementing

1. C Program to Write a Sentence to a File
2. C Program to Read a Line From a File and Display it
3. C Program to Display its own Source Code as Output
4. C program to read name and marks of n number of students from user and store them in a file.
5. C program to write all the members of an array of structures to a file using fwrite(). Read the array from the file and display on the screen.

Up Next

COMMAND LINE ARGUMENTS

```
me@linuxbox:~$ 
```