**Files in C**

A file is a container in computer storage devices used for storing data.

**Why files are needed?**

* When a program is terminated, the entire data is lost. Storing in a file will preserve your data even if the program terminates.
* If you have to enter a large number of data, it will take a lot of time to enter them all.  
  However, if you have a file containing all the data, you can easily access the contents of the file using a few commands in C.
* You can easily move your data from one computer to another without any changes.

**Types of Files**

When dealing with files, there are two types of files you should know about:

Text files

Binary files

**1. Text files**

Text files are the normal .txt files. You can easily create text files using any simple text editors such as Notepad.

**2. Binary files**

Binary files are mostly the .bin files in your computer.

Instead of storing data in plain text, they store it in the binary form (0's and 1's).

**File Operations**

In C, you can perform four major operations on files, either text or binary:

1. Creating a new file

2. Opening an existing file

3. Closing a file

4. Reading from and writing information to a file.

**Working with files**

When working with files, you need to declare a pointer of type file. This declaration is needed for communication between the file and the program.

**FILE \*fptr;**

**Opening a file** - for creation and edit

Opening a file is performed using the fopen() function defined in the stdio.h header file.

The syntax for opening a file in standard I/O is:

**fptr = fopen("fileopen","mode");**

For example,

**fopen("E:\\cprogram\\newprogram.txt","w");**

**Opening Modes in Standard I/O**

|  |  |  |
| --- | --- | --- |
| **Mode** | **Meaning of Mode** | **During Inexistence of file** |
| 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. Data is added to the end of the file. | If the file does not exist, 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 exist, it will be created. |

**Closing a File**

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

Closing a file is performed using the fclose() function.

**fclose(fptr);**

Here, fptr is a file pointer associated with the file to be closed.

**Programs**

**//Open a text file and Write integer data**

#include<stdio.h>

void main()

{

FILE \*ptr;

int n=2;

ptr=fopen("file1.txt","w");

fprintf(ptr,"%d",n);

fclose(ptr);

}

**//Open a text file and Write integer,float,char data**

#include<stdio.h>

void main()

{

FILE \*ptr;

int n=2;

float m=2.345;

char var='A';

ptr=fopen("file1.txt","w");

fprintf(ptr,"%d\t%f\t%c\t",n,m,var);

fclose(ptr);

}

**//Open a text file and Write integer,float,char data at runtime**

#include<stdio.h>

void main()

{

FILE \*ptr;

int n;

float m;

char var;

printf("Enter number\n");

scanf("%c%d%f",&var,&n,&m);

if(ptr==NULL)

{

printf("Error\n");

exit(1);

}

ptr=fopen("file1.txt","w");

fprintf(ptr,"%d\t%f\t%c\t",n,m,var);

fclose(ptr);

}

**Static Memory Allocation**

When memory for the program is allocated during compile time, it is called Static Memory Allocation. The compiler allocates the required memory for the program before the execution of the program.

### Example

#include <stdio.h>

int main()

{

  int a;

  int b[10];

  return 0;

}

## Dynamic Memory Allocation

When memory for the program is allocated during execution time, it is called Dynamic Memory Allocation. The compiler allocates the required memory for the program during the execution of the program.



|  |  |  |
| --- | --- | --- |
| **S.No** | **Static Memory Allocation** | **Dynamic Memory Allocation** |
| 1 | Memory Allocation is made during the Compilation stage. | Memory Allocation is done during the Execution stage. |
| 2 | Memory of the variables are allocated permanently until the program or function call completes. | The variables' memory is allocated only when required and called by the calloc()/malloc() function. |
| 3 | Allocation is done from the Stack Memory. | Allocation is done from the Heap Memory. |
| 4 | In Static memory, if the memory is allocated for a program, the memory size cannot be changed. | In Dynamic memory, if the memory is allocated for a program, the memory size can be changed later. |
| 5 | Less Efficient Memory Management. | More Efficient memory management.. |
| 6 | Memory allocated cannot be reused. | Allocated memory can be released and used again if not required anymore. |
| 7 | Execution of the program is faster than when the memory is allocated dynamically. | Execution of the program is slower than when the memory is allocated statically.. |
| 8 | Can be considered simple compared to dynamic memory allocation. | It a more complex when it comes to declaring multi-dimensional arrays. |
| 9 | Memory declared stays from the beginning to the end of the program execution. | Memory declared can be freed ans reused, and other memory can be allocated. |
| 10 | Used for arrays | Used for Linked-Lists |

**Dynamic Memory Allocation**

* The concept of dynamic memory allocation in C language enables the C programmer to allocate memory at runtime.
* Dynamic memory allocation in c language is possible by 4 functions of stdlib.h header file.
  + malloc( )
  + calloc( )
  + realloc( )
  + free( )

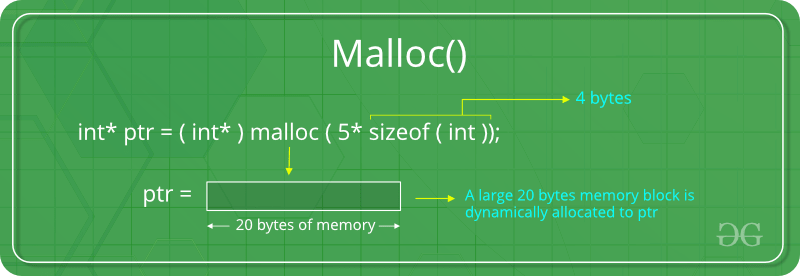
**malloc( )**

* **“malloc”** or **“memory allocation”** method in C is used to dynamically allocate a single large block of memory with the specified size.
* Returns a pointer of type void which can be cast into a pointer of any form.

**ptr = (cast-type\*) malloc(byte-size);**

**Example:**

* + ptr = (int\*) malloc(100 \* sizeof(int));
  + Since the size of int is 4 bytes, malloc will allocate 400 bytes of memory.
  + However, if the space is insufficient, allocation fails and returns a NULL pointer.



**Program**:

#include<stdio.h>

int main()

{

int i, n,\*p;

printf("Enter total number of values\n");

scanf("%d",&n);

p=(int\*)malloc(n\*sizeof(int));

if(p==NULL)

{

printf("unable to allocate memory\n");

exit(0);

}

else

printf("Enter values\n");

for(i=0;i<n;i++)

{

scanf("%d",(p+i));

}

printf("entered numbers are\n");

for(i=0;i<n;i++)

{

printf("%d\t",\*(p+i));

}

free(p);

return 0;

}

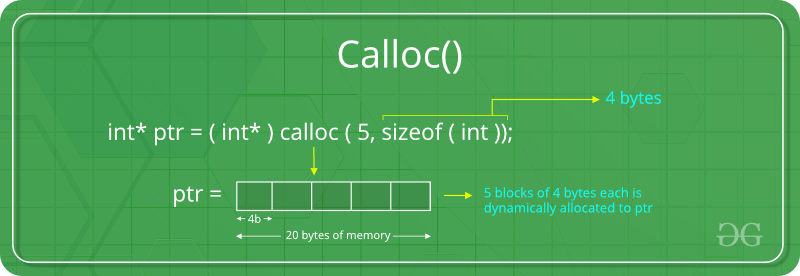
**Calloc**

* **“calloc”** or **“contiguous allocation”** method in C is used to dynamically allocate the specified number of blocks of memory of the specified type. It initializes each block with a default value ‘0’.

**ptr = (cast-type\*)calloc(n, element-size);**

Example:

* + ptr = (float\*) calloc(25, sizeof(float));
  + This statement allocates contiguous space in memory for 25 elements each with the size of the float
  + If space is insufficient, allocation fails and returns a NULL pointer.



realloc

* The function realloc use to change the size of the memory block and it alter the size of the memory block without loosing the old data, it is called reallocation of memory.

ptr = realloc(ptr, x);