system() in C/C++

exec family of functions in C

Difference between fork() and exec()

Check if given Preorder, Inorder and Postorder traversals are of same tree | Set 2

Difference between pointer to an array and array of pointers

Count substrings that contain all vowels | SET 2

C program to sort an array using pointers

Basic Code Optimizations in C.

dot (.) operator in C/C++

Features and Use of Pointers in C/C++

How can we use Comma operator in place of curly braces?

OpenMP | Hello World program

Difference between while and do-while loop in C, C++, Java

Sum of an array using MPI

__builtin_inf() functions of GCC compiler

C Program to count the Number of

exec family of functions in C

The exec family of functions replaces the current running process with a new process. It can be used to run a C program by using another C program. It comes under the header file **unistd.h.** There are many members in the exec family which are shown below with examples.

execvp : Using this command, the created child process does not have to run the same program as the parent process does. The **exec** type system calls allow a process to run any program files, which include a binary executable or a shell script . **Syntax:**

```
int execvp (const char *file, char *const argv[]);
```

file: points to the file name associated with the file being executed.
argv: is a null terminated array of character pointers.

Let us see a small example to show how to use execvp() function in C. We will have two .C files , **EXEC.c** and **execDemo.c** and we will replace the execDemo.c with EXEC.c by calling execvp() function in execDemo.c .

```
//EXEC.c

#include<stdio.h>
#include<unistd.h>

int main()
{
    int i;

    printf("I am EXEC.c called by execvp() ");
    printf("\n");

    return 0;
}
```

Now,create an executable file of EXEC.c using command

```
gcc EXEC.c -o EXEC
```

```
//execDemo.c

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

Characters in a File

time.h header file in C with Examples

scanf("%[^\n]s", str) Vs gets(str) in C with Examples

AKTU (UPTU) Previous Year Solved Papers | C Programming

Constants vs Variables in C language

Analyzing BufferOverflow with GDB

C program to Insert an element in an Array

Types of Literals in C/C++ with Examples

Conditional or Ternary Operator (?:) in C/C++

Difference between C and C#

Now, create an executable file of execDemo.c using command

```
gcc execDemo.c -o execDemo
```

After running the executable file of execDemo.cby using command ./excDemo, we get the following output:

```
I AM EXEC.c called by execvp()
```

When the file execDemo.c is compiled, as soon as the statement execvp(args[0],args) is executed, this very program is replaced by the program EXEC.c. “Ending——” is not printed because because as soon as the execvp() function is called, this program is replaced by the program EXEC.c.

execv : This is very similar to execvp() function in terms of syntax as well. The syntax of **execv()** is as shown below:**Syntax:**

```
int execv(const char *path, char *const argv[]);
```

path: should point to the path of the file being executed.
argv[]: is a null terminated array of character pointers.

Let us see a small example to show how to use execv() function in C. This example is similar to the example shown above for execvp() . We will have two .C files , **EXEC.c** and **execDemo.c** and we will replace the execDemo.c with EXEC.c by calling execv() function in execDemo.c .

```
//EXEC.c
#include<stdio.h>
#include<unistd.h>

int main()
{
    int i;

    printf("I am EXEC.c called by execv() ");
    printf("\n");
    return 0;
}
```

Now,create an executable file of EXEC.c using command

```
gcc EXEC.c -o EXEC
```

time.h localtime() function in C with Examples

asctime() and asctime_s() functions in C with Examples

return statement in C/C++ with Examples

size of char datatype and char array in C

Now, create an executable file of execDemo.c using command

```
gcc execDemo.c -o execDemo
```

After running the executable file of execDemo.c by using command ./excDemo, we get the following output:

```
I AM EXEC.c called by execv()
```

execlp and execl : These two also serve the same purpose but the syntax of them are a bit different which is as shown below:**Syntax:**

```
int execlp(const char *file, const char *arg,.../* (char  *) NULL
int execl(const char *path, const char *arg,.../* (char  *) NULL *
```


file: file name associated with the file being executed

const char *arg and ellipses : describe a list of one or more pointers to null-terminated strings that represent the argument list available to the executed program.

The same C programs shown above can be executed with execlp() or execl() functions and they will perform the same task i.e. replacing the current process the with a new process.

- **execvpe and execle** : These two also serve the same purpose but the syntax of them are a bit different from all the above members of exec family. The synatxes of both of them are shown below :
  Syntax:

```
int execvpe(const char *file, char *const argv[],char *const envp[]);
```

```
Syntax:
int execle(const char *path, const char *arg, .../*, (char *) NULL,
char * const envp[] */);
```

The syntaxes above shown has one different argument from all the above exec members, i.e.

char * const envp[]: allow the caller to specify the environment of the executed program via the argument envp.
envp: This argument is an array of pointers to null-terminated strings and must be terminated by a null pointer. The other functions take the environment for the new process image from the external variable environ in the calling process.

Reference: exec(3) man page

This article is contributed by **MAZHAR IMAM KHAN**. If you like GeeksforGeeks and would like to contribute, you can also write an article using contribute.geeksforgeeks.org or mail your article to contribute@geeksforgeeks.org. See your article appearing on the GeeksforGeeks main page and help other Geeks.

Please write comments if you find anything incorrect, or you want to share more information about the topic discussed above.

Recommended Posts:

Difference between fork() and exec()
Functions in C/C++
Nested functions in C
Thread functions in C/C++
C | Functions | Question 2
C | Functions | Question 3
C | Functions | Question 4
C | Functions | Question 5
C | Functions | Question 10
C | Functions | Question 9
C | Functions | Question 8
C | Functions | Question 7
C | Functions | Question 6
Macros vs Functions
C | Functions | Question 11

Improved By : AakashkumarGoryan, ShubhamMaurya3

Article Tags : C C-Library

Practice Tags : C

13

4.1

To-do Done

Based on **17** vote(s)

Feedback/ Suggest Improvement | Add Notes | Improve Article

Please write to us at contribute@geeksforgeeks.org to report any issue with the above content.

Writing code in comment? Please use ide.geeksforgeeks.org, generate link and share the link here.

Load Comments

A computer science portal for geeks

5th Floor, A-118,
Sector-136, Noida, Uttar Pradesh - 201305
feedback@geeksforgeeks.org