

	-5%	-5%	
₹44	₹784	₹535	₹90

Program to replace process image using execl()

[Leave a Comment](#) / [Programs](#) / By [Baljit Singh Saini](#)

There is a family of function which can be used for replacing the current process with a new process. They differ in the number of arguments and the way they start a new process. The various functions are *execl*, *execlp*, *execle*, *execvp* and *execvp*. In this post we will write a Program to replace process image using *execl*()

We will discuss the use of *execl* and the others can be used on similar lines to replace process image. The syntax for *execl* is

```
int execl(const char *path, const char *arg0, ..., (char *)0);
```

The 1st argument is the PATH for the program *while the last argument will be NULL*.

Program: Program to replace process image using *execl*(). Process *ps* will replace the image of current process

```
#include<stdio.h>
#include<unistd.h>
int main()
{
    printf("Before execl\n");
    execl("/bin/ps", "ps", "-a", NULL); //
    printf("After execlp\n");
}
```

Output

```
$gcc -o exec execl.c

$./exec

Before execl
PID TTY      TIME CMD
122 tty1    00:00:00 ps
```

How it Works?

The program prints the first message and then calls *execl*. *execl* searches for the program *ps* in the *PATH* variable and executes it replacing the original program. Hence the second message doesn't get printed. Also, It can be seen in the output that the name of the process running is *ps* and not the user process *exec*.

ivame Ankle Length Layering Skirt
skin

This Layering Skirt is designed to give you discrn
through/mesh skirts and dresses. Wear it under
create an outfit that screams confidence.Can be
and dresses.Length: Knee-length

How to Replace Process image in Linux || `exec()` || Program



Viva Questions on Program to replace process image using `exec()`

Q1. `exec()` function returns only if _____

Q2. The statements that appear after `exec()` function also executes – True/False.

Q3. How is `exec()` function different from `system()` function?

Relevant Programs

[Program to create a child process using `fork\(\)` system call](#)

[Process Synchronization using Semaphores](#)

[Inter-process communication using `pipe\(\)` function](#)

[Difference between `exec\(\)` and `system\(\)` functions](#)

[← Previous Post](#)

[Next Post →](#)

Leave a Comment

You must be [logged in](#) to post a comment.

Search ...



OS Lab

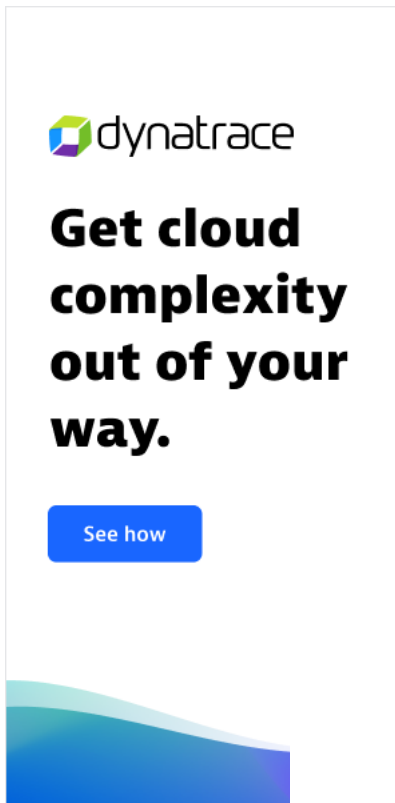
Program to create Threads in



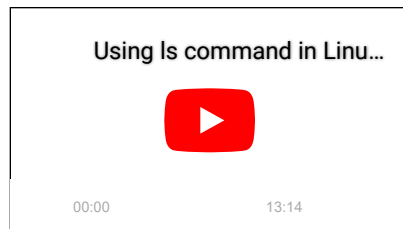
00:00

09:29

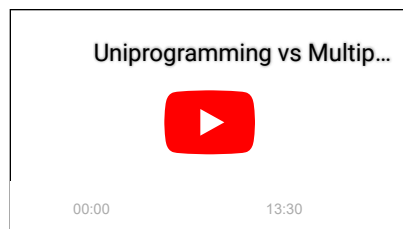




Linux Essentials Series



OS Theory



Categories

[Database](#)

[Entity-Relationship\(ER\)](#)

[Introduction to Database](#)

[Linux](#)

▼ [commands](#)

[Deadlock](#)

[Disk Management](#)

[File Management](#)

[Introduction](#)

[IPC](#)

[Memory Management](#)

[Practice Problems](#)

[Process](#)

[Process Synchronization](#)

[System calls](#)

[Thread](#)

[Programs](#)

[Question Hub](#)

[UGC-NET MCQ's for OS](#)

[Uncategorized](#)

Copyright © 2023 Dextutor | Powered by [Astra WordPress Theme](#)

