```
int old_fd, new_fd;
old_fd=open("test.txt",O_RDWR);
printf("File descriptor is %d\n",old_fd);
new_fd=dup(old_fd);
printf("New file descriptor is %d\n",new_fd);
```

Program for dup() system call

Leave a Comment / Programs / By Baljit Singh Saini

Before starting with the program to use dup() system call in linux, let's first understand the use and syntax of dup()

Use

dup() system call is used to duplicate a file descriptor. It creates a copy of the old file descriptor to a new file descriptor. The new file descriptor can be system-generated or a number of your choice depending upon either you use dup() or dup2().

Syntax

```
#include<unistd.h>
int dup(int oldfd);
int dup2(int oldfd, int newfd);
```

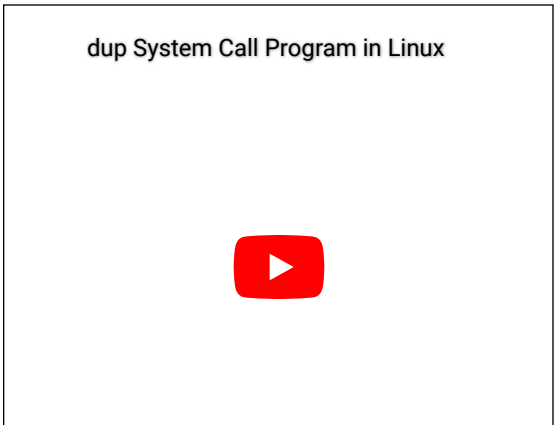
dup() takes the old file descriptor as the input parameter. On success, it returns the lowest-numbered unused file descriptor as the new file descriptor.
Now, we can use the old and the new file descriptor because they refer to the same file.
Note: Both these file descriptor share the same file offset i.e., if you use old file descriptor to read from the file and the file pointer is positioned at 10th position. Even if you use new file descriptor the reading/writing within the file will start from 10th position only.

dup2() system call is same as dup() with the difference that here you can give the value of new file descriptor of your own choice. If the new file descriptor is already open, it is silently closed and then reopened for reuse.

Programs

Program 1: Program for dup() system call in C to duplicate a file descriptor.

```
{
int old_fd, new_fd;
old_fd=open("test.txt",O_RDWR);
printf("File descriptor is %d\n",old_fd);
new_fd=dup(old_fd);
printf("New file descriptor is %d\n",new_fd);
}
```



How it works?

open() system call returns the file descriptor of the file "test.txt". The old_fd variable stores the value of this file descriptor. Next, we duplicate the file descriptor by using dup(), which assign the lowest unused file descriptor value. Finally, the new_fd variable stores the value of the new file descriptor. Now, both old_fd and new_fd point to the same file "test.txt"

Output

Program 2: Program to use dup2() system call in linux to duplicate a file descriptor.

```
//dup2.c
#include<unistd.h>
#include<stdio.h>
3include<fcntl.h>
int main()
{
int old_fd, new_fd;
old_fd=open("test.txt",O_RDWR);
printf("File descriptor is %d\n",old_fd);
new_fd=dup2(old_fd,7);
printf("New file descriptor is %d\n",new_fd);
}
```

Program 3: Program to show that both file descriptor point to the same file and same pointer position is maintained

```
//create a file test.txt with the content "1234567890abcdefghij54321"
#include<unistd.h>
#include<stdio.h>
#include<fcntl.h>
int main()
{
int old_fd, new_fd;
char buff[10];
old_fd=open("test.txt",O_RDWR);
read(old_fd,buff,10);//read first 10 characters using old file descriptor
write(1,buff,10);//prints them on screen
new_fd=dup(old_fd);//duplicates file descriptor
read(old_fd,buff,10);//this read will read the next 10 characters even if new file descriptor is used
write(1,buff,10);
}
```

How it works?

The file "test.txt" is opened with the pointer positioned at '1'. The *read()* system call reads the first 10 characters i.e., 1234567890 and write() prints them on screen. The call to read() leaves the pointer pointing to 'a' (the 11th character of the file). Next, the file descriptor is duplicated and read() is used to read 10 more characters. Since, the pointer is at 'a' it reads the next 10 characters starting 'a' which are "abcdefghij". Hence, the output "1234567890abcdefghij"

Output

Practice Programs on dup() system call in C

Q1. Write a program using dup()/dup2() to assign '1' as the duplicate file descriptor.
Q2. Write a program to duplicate a file descriptor of a file. Use the old file descriptor to read the first 5 characters and the new file descriptor to append some new content to the file.

Viva questions on dup() system call

Q1. Can you assign 0,1,2 as new file descriptors using dup()?
Q2. Can you assign 0,1,2 as new file descriptors using dup2()?

Relevant Programs

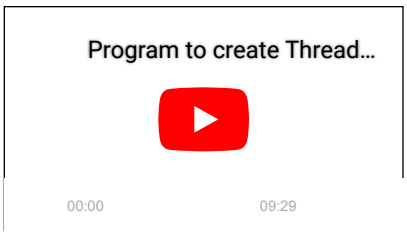
- open() system call
- lseek() system call
- write()/read() system call

Leave a Comment

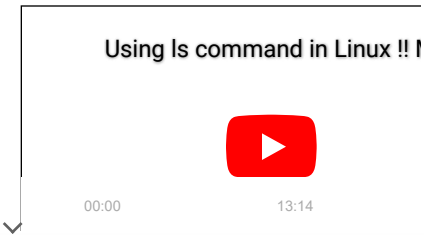
You must be logged in to post a comment.

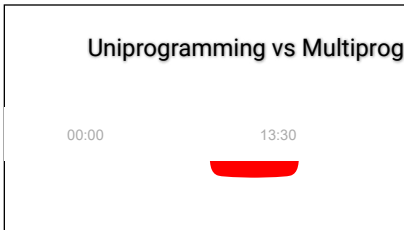
Search ...

OS Lab



Linux Essentials Series





Categories

UGC-NET MCQ's for OS

Uncategorized