How to Copy Files in Linux and Unix? 10 cp Command Examples

thegeekstuff.com/2013/03/cp-command-examples/

cp is one of the basic command in Unix. You already know that it is used to copy one or more files or directories from source to destination.

While this tutorial is for beginners, it is also helpful for everybody to quickly review various cp command options using some practical examples.

Even if you are using cp command all the times, probably one or more examples explained below might be new to you.

The general form of copy command:

```
cp [option] source
destination
```

1. Copy a file or directory from source to destination

To copy a file, you need to pass source and destination to the copy command. The following example copies the file from project/readme.txt to projectbackup/readme-new.txt

```
$ cp project/readme.txt projectbackup/readme-
new.txt

$ cd projectbackup/

$ ls
readme-new.txt
```

If you want to copy a file from one folder to another with the same name, just the destination directory name is good enough as shown below.

```
$ cp project/readme.txt
projectbackup/

$ cd projectbackup/

$ ls
readme.txt
```

A directory (and all its content) can be copied from source to destination with the recursive option -r as shown below:

```
$ ls project
src/  bin/  doc/  lib/  test/  readme.txt
LICENSE

$ cp -r project/ backup/

$ ls backup
src/  bin/  doc/  lib/  test/  readme.txt
LICENSE
```

2. Copy multiple files or directories

You can copy more than one file from source to destination as shown below:

```
$ cd src/
$ cp global.c main.c parse.c
/home/thegeekstuff/projectbackup/src/
```

If the source files has a common pattern, use wild-cards as shown below. In this example, all c extension files gets copied to /home/thegeekstuff/projectbackup/src/ directory.

```
$ cp *.c
/home/thegeekstuff/projectbackup/src/
```

Copy multiple directories as shown below.

```
$ cd project/

$ cp -r src/ bin/
/home/thegeekstuff/projectbackup/
```

3. Backup before copying into a destination

In case if the destination file is already present with the same name, then cp allows you to backup the destination file before overwriting it.

In this example, the readme.txt exists in both project/ and projectbackup/ directory, and while copying it from project/ to projectbackup/, the existing readme.txt is backed up as shown below:

```
$ cd projectbackup

$ ls -l readme.txt
-rw-r--r-- 1 bala geek 1038 Jan  8 13:15 readme.txt

$ cd ../project

$ ls -l readme.txt
-rw-r--r-- 1 bala geek 1020 Jan  8 12:25 readme.txt

$ cp --backup readme.txt
/home/thegeekstuff/projectbackup/
```

The existing file has been moved to readme.txt~ and the new file copied as readme.txt as shown below.

```
$ cd /home/thegeekstuff/projectbackup/
$ ls -l
-rw-r--r-- 1 bala geek 1020 Jan  8 13:36 readme.txt
-rw-r--r-- 1 bala geek 1038 Jan  8 13:15
readme.txt~
```

Talking about backup, it is important for you to understand how rsync command works to backup files effectively.

4. Preserve the links while copying

When you execute the cp command, if the source is a link file, then the actual file gets copied and not the link file. In case if you only want to copy the link as it is, specify option -d as shown below:

The following shows that without option -d, it will copy the file (and not the link):

```
$ cd project/bin

$ ls -l startup.sh
lrwxrwxrwx 1 root root 18 Jan  8 13:59 startup.sh ->
../test/startup.sh

$ cp startup.sh /home/thegeekstuff/projectbackup/bin/

$ cd /home/thegeekstuff/projectbackup/bin/

$ ls -l
-rw-r--r--  1 root root       102 Jan  8 14:02 startup.sh
```

To preserve the link while copying, do the following:

```
$ cd project/bin

$ cp -d startup.sh /home/thegeekstuff/projectbackup/bin/

$ ls -l startup.sh
lrwxrwxrwx 1 root root 18 Jan  8 14:10 startup.sh ->
../test/startup.sh
```

5. Don't overwrite an existing file

If you want to copy only when the destination file doesn't exist, use option -n as shown below. This won't overwrite the existing file, and cp command will return with success exit code as shown below:

```
$ cd projectbackup

$ ls -l readme.txt
-rw-r--r-- 1 bala geek 1038 Jan  8 13:15 readme.txt

$ cd ../project

$ ls -l readme.txt
-rw-r--r-- 1 bala geek 1020 Jan  8 12:25 readme.txt

$ cp -n readme.txt
/home/thegeekstuff/projectbackup/bin/

$ echo $?
0
```

As you see below, the destination file didn't get overwritten.

```
$ cd projectbackup

$ ls -l readme.txt
-rw-r--r-- 1 bala geek 1038 Jan  8 13:15
readme.txt
```

6. Confirm before overwriting (interactive mode)

When you use -i option, it will ask for confirmation before overwriting a file as shown below.

```
$ cp -i readme.txt /home/thegeekstuff/projectbackup/
cp: overwrite `/home/thegeekstuff/projectbackup/readme.txt'?
y
```

7. Create hard link to a file (instead of copying)

When you execute cp command, it is possible to create a hard link of the file (instead of copying the file). The following example creates the hard link for sample.txt file into directory test/,

```
$ ls -li sample.txt
10883362 -rw-r--r-- 2 bala geek   1038 Jan  9 18:40 sample.txt

$ cp -l sample.txt test/

$ ls -li test/sample.txt
10883362 -rw-r--r-- 2 bala geek   1038 Jan  9 18:40
test/sample.txt
```

As seen above, the test/sample.txt is a hard linked file to sample.txt file and the inode of both files are the same.

8. Create Soft link to a file or directory (instead of copying)

When you execute cp command, it is possible to create a soft link to a file or directory. In the following example, a symbolic link gets created for libFS.so.6.0.0 as libFS.so,

```
# cd /usr/lib/

# ls -l libFS.so.6.0.0
-rw-r--r-- 1 root root 42808 Nov 19  2010 libFS.so.6.0.0

# cp -s libFS.so.6.0.0 libFS.so

# ls -l libFS.so
lrwxrwxrwx 1 root root 14 Jan  9 20:18 libFS.so ->
libFS.so.6.0.0
```

9. Preserve attributes of file or directory while copying

Using -p option, you can preserve the properties of a file or directory as shown below:

```
$ ls -l sample.txt
-rw-r--r-- 2 bala geek   1038 Jan  9 18:40 sample.txt

$ cp -p sample.txt test/

$ ls -l test/sample.txt
-rw-r--r-- 2 bala geek   1038 Jan  9 18:40
test/sample.txt
```

It is also possible to preserve only the required properties like mode, ownership, timestamps, etc.,

The following example preserves the mode of a file while copying it:

```
$ cp --preserve=mode sample.txt
test/
```

10. Copy only when source file is newer than the destination or missing

Copy doesn't take much time for a small file, but it may take considerable amount of time when a huge file is copied. So, while copying a big file, you may want to make sure you do it only when the source file is newer than the destination file, or when the destination file is missing using the option -u as shown below.

In this example, the two files LICENSE and readme.txt will be copied from project/ to projectbackup/. However, the LICENSE file already exists in projectbackup/ directory and that is newer than the one in the project/ directory.

```
$ cd project/

$ ls -l LICENSE readme.txt
-rw-r--r-- 1 bala geek 108 Jan  8 13:14 LICENSE
-rw-r--r-- 1 bala geek 32 Jan  8 13:16 readme.txt

$ cd /home/thegeekstuff/projectbackup/

$ ls -l LICENSE readme.txt
ls: cannot access readme.txt: No such file or
directory
-rw-r--r-- 1 root root 112 Jan  9 20:31 LICENSE
```

So, in this example, there is no need to copy LICENSE file again to projectbackup/ directory. This is automatically taken care by cp command, if you use -u option as shown below. In the below example, only readme.txt file got copied as indicated by the time-stamp on the file.

```
$ cp -u -v LICENSE readme.txt
/home/thegeekstuff/projectbackup/
`readme.txt' -> `/home/thegeekstuff/projectbackup/readme.txt'

$ cd /home/thegeekstuff/projectbackup/

$ ls -l LICENSE readme.txt
-rw-r--r-- 1 bala geek  112 Jan  9 20:31 LICENSE
-rw-r--r-- 1 bala geek   32 Jan  9 22:17 readme.txt
```