8 Practical Examples of Linux Xargs Command for Beginners

The Linux **xargs** command may not be a hugely popular command line tool, but this doesn't take away the fact that it's extremely useful, especially when combined with other commands like find and grep. If you are new to xargs, and want to understand its usage, you'll be glad to know that's exactly what we'll be doing here.

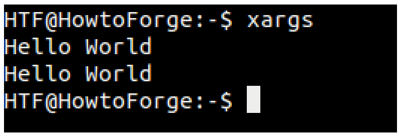
Before we proceed, please keep in mind that all the examples presented in this tutorial have been tested on Ubuntu 14.04 LTS. Shell used is Bash, and version is 4.3.11.

On this page

1. How Xargs command works?

Well, before jumping onto its usage, it's important to understand what exactly Xargs does. In layman's terms, the tool - in its most basic form - reads data from standard input (stdin) and executes the command (supplied to it as argument) one or more times based on the input read. Any blanks and spaces in input are treated as delimiters, while blank lines are ignored.

If no command is supplied as argument to xargs, the default command that the tool executes is echo. For example, in the following example, I just executed 'xargs' and entered 'Hello World' on stdin. As I pressed Ctrl+D (to tell xargs that we're done with the input), the echo command was automatically executed, and 'Hello World' was printed again.



```
HTF@HowtoForge:-$ xargs
Hello World
Hello World
HTF@HowtoForge:-$
```

2. How to use xargs with another command?

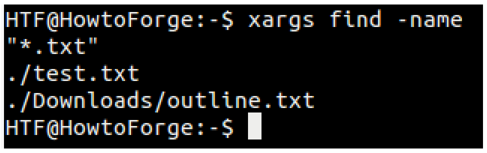
While echo is the default command xargs executes, you can explicitly specify any other command. For example, you can pass the find command along with its '-name' option as argument to xargs, and then pass the name of the file (or type of files) you want find to search as input through stdin.

Here's the complete command in question:

```
xargs find -name
```

For example, we provided "*.txt" in input through stdin, which means we want the find command to search all .txt files in the current directory (as well as its subdirectories).

Here's the command in action:

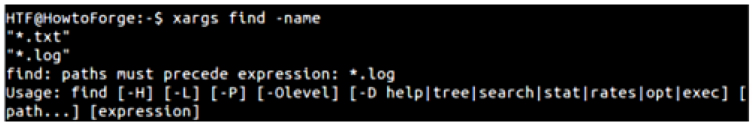


```
HTF@HowtoForge:-$ xargs find -name
"*.txt"
./test.txt
./Downloads/outline.txt
HTF@HowtoForge:-$
```

3. How to make xargs execute command multiple times (once for each input line/argument)

In the example we discussed in the previous section, we wanted to search for .txt files, so we provided "*.txt" in the input. But what if the requirement is to also search for other types of files as well, like .log and .tmp?

Try giving these as input to xargs after "*.txt" and see what happens. Well, you should get an error similar to the following:



```
HTF@HowtoForge:-$ xargs find -name
"*.txt"
"*.log"
find: paths must precede expression: *.log
Usage: find [-H] [-L] [-P] [-Olevel] [-D help|tree|search|stat|rates|opt|exec] [
path...] [expression]
```

That's because xargs passes "*.txt" and "*.log" as file name input to the find command and then find fails as it runs with two file names in input. You can think of the find command running in the following way:

```
find -name "*.txt" "*.log"
```

And it's is not the correct way.

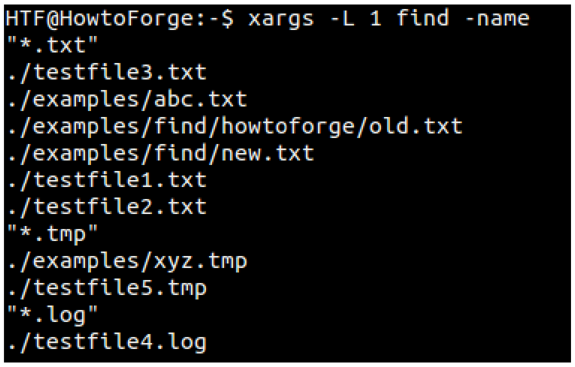
What we want is that the xargs command should first pass "*.txt" to the find command, so that find gives us results related to .txt files, and then "*.log" is passed. This can be done using the -L command line option xargs provides.

The -L command line option requires a number which it treats as the maximum number of non-blank lines that should be passed as input to the command in one time. So, in our case, that value will be 1, as we want one input line to passed as input to find at one time.

So, here's the command that we should run:

```
xargs -L 1 find -name
```

The following screenshot shows the aforementioned command in action:

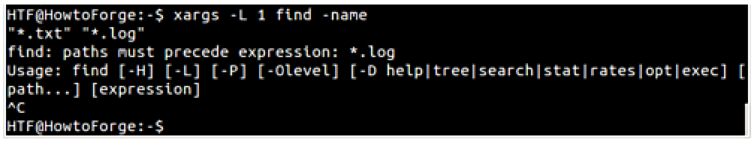


```
HTF@HowtoForge:-$ xargs -L 1 find -name
"*.txt"
./testfile3.txt
./examples/abc.txt
./examples/find/howtoforge/old.txt
./examples/find/new.txt
./testfile1.txt
./testfile2.txt
"*.tmp"
./examples/xyz.tmp
./testfile5.tmp
"*.log"
./testfile4.log
```

So far, so good. But what if you pass the input in the following way:

```
"*.txt" "*.log"
```

Well, the command will fail because '-L 1' will make sure the complete line is passed to the find command.

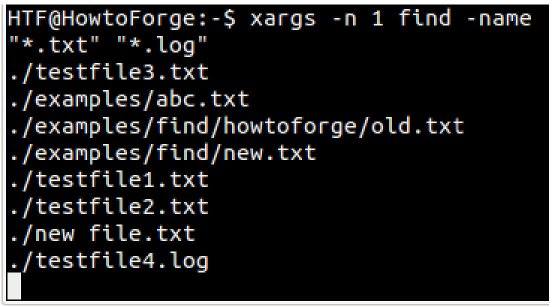


```
HTF@HowtoForge:-$ xargs -L 1 find -name
"*.txt" "*.log"
find: paths must precede expression: *.log
Usage: find [-H] [-L] [-P] [-Olevel] [-D help|tree|search|stat|rates|opt|exec] [
path...] [expression]
^C
HTF@HowtoForge:-$
```

In situations like these, you can use the '-n' command line option.

The '-n' command line option, like '-L', requires a number that would represent the number of arguments you want xargs to pass per command line. In the case we just discussed, "*.txt" and "*.log" should be different arguments, and since we want them to be passed individually, we'll have to provide '1' as value to '-n'.

```
xargs -n 1 find -name
```



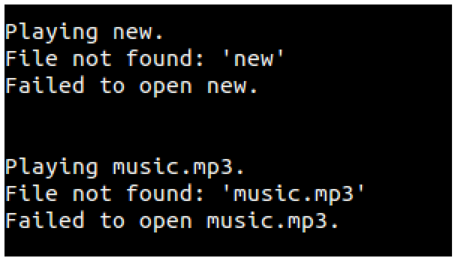
```
HTF@HowtoForge:-$ xargs -n 1 find -name
"*.txt" "*.log"
./testfile3.txt
./examples/abc.txt
./examples/find/howtoforge/old.txt
./examples/find/new.txt
./testfile1.txt
./testfile2.txt
./new file.txt
./testfile4.log
```

So these are the ways in which we can make xargs execute a command multiple times.

4. How to make xargs handle filenames with spaces

If you recall, in the beginning, we mentioned that xargs treats blank spaces (as well as newline characters) as delimiters. This may cause issues in cases where-in the filenames being passed as input to xargs contain spaces.

For example, suppose the current directory contains a file named 'new music.mp3', and you want xargs to pass this name to mplayer so that the latter can play the file. If you go by the conventional way, you'll get an error:



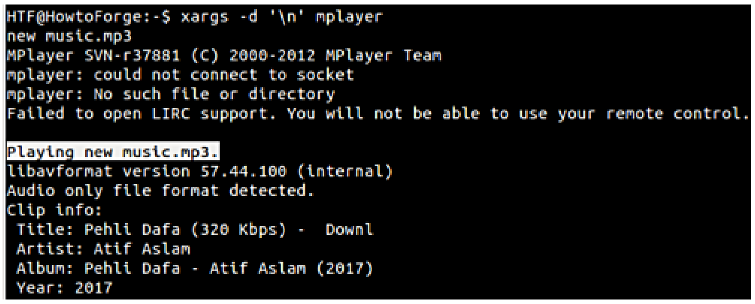
```
Playing new.
File not found: 'new'
Failed to open new.


Playing music.mp3.
File not found: 'music.mp3'
Failed to open music.mp3.
```

That's because due to the space between 'new' and 'music.mp3', these are treated as two different files by mplayer. One way to solve this problem is to use backslash while providing the input:

```
new\ music.mp3
```

The other way is to change the delimiter, which you can do using the -d option. For example, you can ask xargs to only consider newline as a delimiter, something which can be done in the following way:

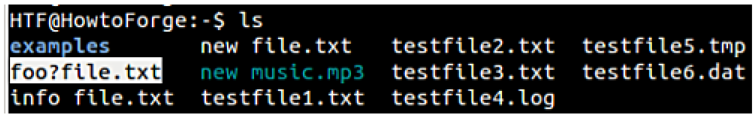
```
xargs -d '\n' mplayer
```



```
HTF@HowtoForge:-$ xargs -d '\n' mplayer
new music.mp3
MPlayer SVN-r37881 (C) 2000-2012 MPlayer Team
mplayer: could not connect to socket
mplayer: No such file or directory
Failed to open LIRC support. You will not be able to use your remote control.

Playing new music.mp3.
libavformat version 57.44.100 (internal)
Audio only file format detected.
Clip info:
 Title: Pehli Dafa (320 Kbps) -  Downl
 Artist: Atif Aslam
 Album: Pehli Dafa - Atif Aslam (2017)
 Year: 2017
```

5. How to make xargs handle filenames with newline characters

So, now we know how we can make xargs handle filenames that contain white spaces. But what about newlines? Yes, filenames can have newline characters as well. For example, there's a file named 'foo'[newline]'file.txt'. On terminal, the name is displayed as 'foo?file.txt'.



```
HTF@HowtoForge:-$ ls
examples        new file.txt   testfile2.txt  testfile5.tmp
foo?file.txt    new music.mp3  testfile3.txt  testfile6.dat
info file.txt   testfile1.txt  testfile4.log
```

Now, suppose, I want the find command to find all files with names starting with text 'foo', and then use xargs to open these files in gedit. Here's the command:

```
find . -name "foo*" | xargs gedit
```

Sadly, the command fails to achieve its purpose as a new file with name 'foo' opens in Gedit. That's likely because the newline character is treated as a delimiter by xargs.

To sort this problem out, use the -print0 option that find offers and couple it with xarg's -0 option. Here's how the man page of find explains this option:

```
 -print0
True; print the full file name on the standard output, followed by a null character (instead of thenewline charac
ter that -print uses). This allows file names that contain newlines or other types ofwhite space to be correctly
 interpreted by programs that process the find output. This option corresponds to the -0 option of xargs.
```

So, here's the command we should be using:

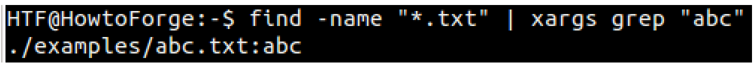
```
find . -name "foo*" -print0 | xargs -0 gedit
```

6. How to find files containing a specific text

The xargs command comes in really handy if you want to search for files containing a specific text/string. For example, if you want to find .txt files containing text 'abc', then this can be done in the following way:

```
find -name "*.txt" | xargs grep "abc"
```

Here's the command in action:



```
HTF@HowtoForge:-$ find -name "*.txt" | xargs grep "abc"
./examples/abc.txt:abc
```

As is quite obvious, in this case, the xargs command gets its input from the find command.

7. How to make xargs accept input from a file

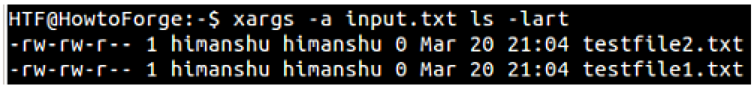
So far, we have seen xargs accepting input from stdin (which is the default behavior) as well as from another command (discussed in last section). However, if you want, you can also have xargs accept input from a file.

The command's -a option lets you do this. This option expects a filename - this would be the file from which xargs will be reading input.

For example, suppose there's a file with name 'input.txt'. It contains the input for xargs (basically, the contents are two filenames: testfile1.txt and testfile2.txt). Here's how xargs can be used to read input from this file, and pass it to another command, say ls.

```
xargs -a input.txt ls -lart
```

Here's the command in action:

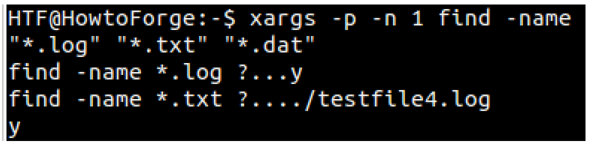


```
HTF@HowtoForge:-$ xargs -a input.txt ls -lart
-rw-rw-r-- 1 himanshu himanshu 0 Mar 20 21:04 testfile2.txt
-rw-rw-r-- 1 himanshu himanshu 0 Mar 20 21:04 testfile1.txt
```

8. How to make xargs seek user permission before executing a command

We have already discussed how you can make xargs execute a command multiple times (see point 3 above). But what if you want xargs to ask for your permission each time it runs the command? Is that possible? Yes, it is - the -p command line option makes it possible.

Here's an example of how you can run xargs in interactive mode:



```
HTF@HowtoForge:-$ xargs -p -n 1 find -name
"*.log" "*.txt" "*.dat"
find -name *.log ?...y
find -name *.txt ?..../testfile4.log
y
```

Conclusion

We've just scratched the surface here as there are plenty of other features the xargs command provides in the form of command line options. Do try the xargs concepts/features we've explained in this tutorial, and in case of any doubt or query, go through its man page.

Suggested articles

0 Comment(s)

Add comment

Name *

Email *

p

I'm not a robot

reCAPTCHA

Privacy - Terms

Submit comment

Tutorials > **8 Practical Examples of Linux Xargs Command for Beginners**

Sign up now!

Tutorial Info

Author:	Himanshu Arora
Published:	Mar 27, 2017
Tags:	linux, shell

Share This Page

Tweet Follow 27.8K followers

Recommend 18

G+1 5