By using `-n` along with the previous command, we can split the input into multiple lines having two words each as follows:

```
$ echo "splitXsplitXsplitXsplit" | xargs -d X -n 2
split split
split split
```

There's more...

We have learned how to format `stdin` to different output as arguments from the previous examples. Now, let's learn how to supply this formatted output as arguments to commands.

Passing formatted arguments to a command by reading stdin

Write a small custom `echo` script for better understanding of example usages with `xargs` to provide command arguments:

```
#!/bin/bash
#Filename: cecho.sh

echo $*'#'
```

When arguments are passed to the `cecho.sh` shell, it will print the arguments terminated by the # character. For example:

```
$ ./cecho.sh arg1 arg2
arg1 arg2 #
```

Let's have a look at a problem:

- I have a list of arguments in a file (one argument in each line) to be provided to a command (say, `cecho.sh`). I need to provide arguments in two methods. In the first method, I need to provide one argument each for the command as follows:

  ```
  ./cecho.sh arg1
  ./cecho.sh arg2
  ./cecho.sh arg3
  ```

  Or, alternately, I need to provide two or three arguments each for each execution of the command. For two arguments each, it would be similar to the following:

  ```
  ./cecho.sh arg1 arg2
  ./cecho.sh arg3
  ```

- In the second method, I need to provide all arguments at once to the command as follows:

  ```
  ./cecho.sh arg1 arg2 arg3
  ```

Run the preceding commands and note the output before going through the following section.

These problems can be solved using `xargs`. We have the list of arguments in a file called `args.txt`. The contents are as follows:

```
$ cat args.txt
arg1
arg2
arg3
```

For the first problem, we can execute the command multiple times with one argument per execution, therefore, use:

```
$ cat args.txt | xargs -n 1 ./cecho.sh
arg1 #
arg2 #
arg3 #
```

For executing a command with `X` arguments per each execution, use:

```
INPUT | xargs -n X
```

For example:

```
$ cat args.txt | xargs -n 2 ./cecho.sh
arg1 arg2 #
arg3 #
```

For the second problem, in order to execute the command at once with all the arguments, use:

```
$ cat args.txt | xargs ./ccat.sh
arg1 arg2 arg3 #
```

In the preceding examples, we have supplied command-line arguments directly to a specific command (for example, `cecho.sh`). We could only supply the arguments from the `args.txt` file. However, in real time, we may also need to add a constant parameter with the command (for example, `cecho.sh`), along with the arguments taken from `args.txt`. Consider the following example with the format:

```
./cecho.sh -p arg1 -l
```

In the preceding command execution `arg1` is the only variable text. All others should remain constant. We should read arguments from a file (`args.txt`) and supply it as:

```
./cecho.sh -p arg1 -l
./cecho.sh -p arg2 -l
./cecho.sh -p arg3 -l
```