How to use read command in Linux

Sometimes when interacting with a Linux system, you may need to prompt users to enter data or read data from files, or even want to set a timeout.

You can perform these and many other tasks using the read command and its various options.

This article will guide you through the basics of the read command and its options through some examples.

What is the Read command?

In Linux, you can use the read command to capture user input or read a line from standard input (stdin). This command reads the total number of bytes from the given file descriptor and stores them in the buffer. The command then returns the number of bytes read, zero, or error.

For example, if the number or count is zero, the command will refer to the end of the file. But when successful, the command returns the number of bytes read. If the read command finds some errors, it returns -1.

Before exploring the read command options, let's first look at the syntax of the read command:

```
read [options] [name…]
```

Here, the **options** parameter specifies various flags used to modify the behavior of the read command. Additionally, the **name** parameter specifies the names of multiple variables used to store input data. If no name is provided, input will be retained in the **$REPLY bash variable.**

read command options

Bash's read command has many options for controlling user input. Some options do not need additional parameters, while others do.

Let's explore some of the options we can use with the read command:

Option	Describe
-a	It stores input data as an array instead of separate variables.
-S	Runs silently, meaning input data is not displayed on the terminal
-e	Enables readline library support and reading input lines
-i	Provides the initial input value that appears at the prompt when using readline
-p	Displays the specified prompt before reading the input

Option	Describe
-u	Read from a specified file descriptor instead of standard input (stdin)
-d	Allows specifying an input line separator instead of the default newline character
-t	Set input timeout; If input is not received within this time, the read command will return an error
-r	When set, backslashes are not treated as escape characters
-n	Read only the specified number of characters

Enter the following command to output the read command help menu:

```
head --help
```

How to read input using read command

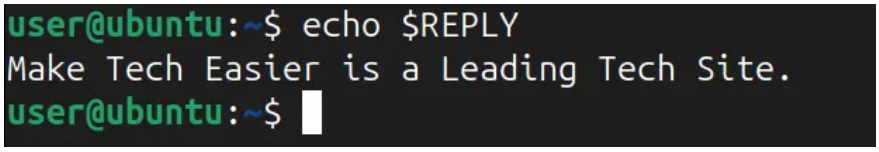
The simplest way to use the read command is to use it without any arguments or options. When executing just the read command, the command will ask you for the input you want to read. After providing input, the command exits and stores that input in a default variable named REPLY.

Let's take the following example:

```
read
```

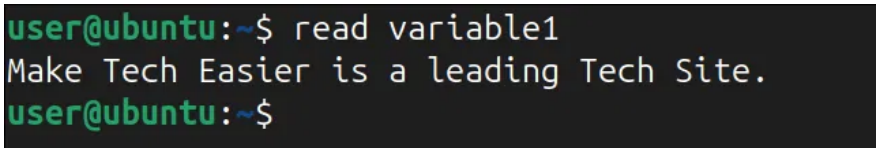
Now, after providing input, display it using echo command:

```
echo $REPLY
```



While reading the input value, you can also store it in any other specific variable. For example, to store the result in a variable, enter the read command followed by the variable name:

```
read variable1
```



Now, to display the results, you need to use the echo command with the variable storing the value:

```
echo $variable1
```

```
user@ubuntu:~$ echo $variable1
Make Tech Easier is a leading Tech Site.
user@ubuntu:~$
```

Read many values

There is no direct way to read multiple values ??using the read command. However, you can split an input sentence into multiple words and store them in different variables.

Consider the following example:

```
read variable1 variable2 variable3
```

```
user@ubuntu:~$ read variable1 variable2 variable3
Make Tech Easier is a tech leading tech site.
user@ubuntu:~$
```

Here, you store the first word of the sentence in the first variable, the second word in the second variable, and all the remaining words in the last variable provided.

Let's return the output with the following command:

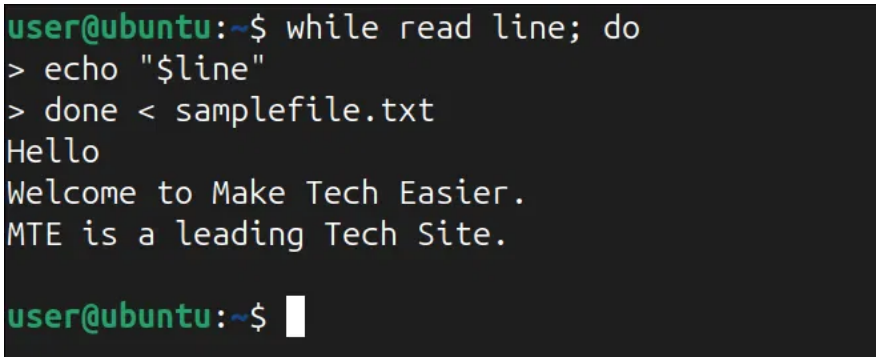
```
echo $variabl_namee>
```

```
user@ubuntu:~$ echo $variable1
Make
user@ubuntu:~$ echo $variable2
Tech
user@ubuntu:~$ echo $variable3
Easier is a tech leading tech site.
user@ubuntu:~$
```

Read from a file

Although the read command is primarily for user input, you can also use it to read lines from a file. To do this, simply use a while loop, an echo command, and a read command followed by the specific variable name:

```
while read line; do echo "$line" done  samplefile.txt
```



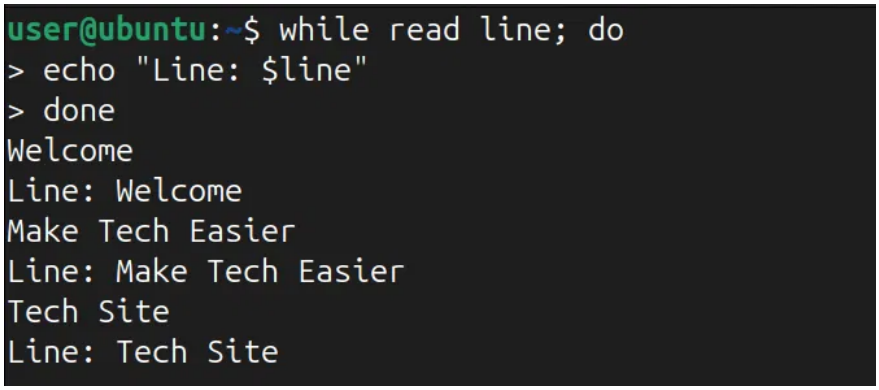
Here, the while loop reads each line of "samplefile.txt" and records it in the line variable. After reading all lines of the file, the echo command will display the value of that line.

Read input in loop

You can also capture user input in a repeating sequence using the read command with a while loop. This is useful when you want to collect multiple inputs or continue until a specific condition is met.

For example, let's read multiple inputs and also display them on the terminal:

```
while read line; do echo "Line: $line" done
```



Furthermore, the loop continues until the end of file (EOF) signal is received, usually by pressing **Ctrl + D** .

You finished reading the article "**How to use read command in Linux**" edited by the **TipsMake** team. We hope this article has provided you with many useful tech tips and tricks. You can search for similar articles on tips and guides. Thank you for reading and for following us regularly.