# Bash Command Line Arguments

Command line arguments are passed to the bash script during its execution. They are mentioned just after the script filename and are separated by space. If any argument has space then you must enclose that argument in single or double quotes.

## How to pass command line arguments to Bash Script?

```
#bash scriptname.sh argument1 argument2 ...
```

```
#./scriptname.sh argument1 argument2 ...
```

| Special Variables | Explanation |
|---|---|
| $0 | It stores the name of the bash script. |
| $1 to $n | It stores the value of command line arguments. `$1` refers to the first argument, `$2` refers to the second argument and so on $n refers to the nth argument. From tenth argument onwards, enclose the number in curly braces like ${10}, ${11}, etc. |
| $* | It stores all the command line arguments as a single word. |
| $@ | It stores all the command line arguments as separate words. You can iterate through arguments using `for` command. |
| $# | It refers to the number of command line arguments. |

## Example: Bash Command Line Arguments

```
echo "Script name: $0"
echo "Total number of command line arguments: $#"
echo "First argument: $1"
echo "Second argument: $2"
echo "Third argument: $3"
```

```
#./main.sh 123 Android "Windows OS"
```

Output of the above program

```
Script name: main.sh
Total number of command line arguments: 3
First argument: 123
Second argument: Android
Third argument: Windows OS
```

## Difference between $* and $@

You might think, if both $* and $@ stores all the command-line arguments then what is the difference between them. Go through the below example to better understand it.

```
count=1
for arg in "$*"
do
  echo "\$* Argument $count: $arg"
  count=$((count+1))
done
count=1
for arg in "$@"
do
  echo "\$@ Argument $count: $arg"
  count=$((count+1))
done
```

Output of the above program

```
#./main.sh Android iOS Windows
```

```
$* Argument 1: Android iOS Windows
$@ Argument 1: Android
$@ Argument 2: iOS
$@ Argument 3: Windows
```

As you can see above, `$*` considers all command line arguments as a single word while `$@` consider them as a separate words.