Passing Arguments to a Bash Script

In Bash scripts, you can pass arguments to the script when you run it from the command line. These arguments can be accessed within the script using special variables.

Accessing Arguments in a Script

When you run a script, the arguments passed to it are stored in **positional parameters**. These are accessible through special variables:

Variable	Description
`$0`	The name of the script itself
`$1`, `$2`, ...	The first, second, third, etc., arguments passed to the script
`$#`	The total number of arguments passed to the script
`$@` or `$*`	All the arguments passed to the script as a list

Basic Example

Create a script (greet.sh) that takes a name as an argument and prints a greeting.

```
#!/bin/bash

# greet.sh - A simple script to greet a user

echo "Hello, $1!"
```

Run the script and pass an argument:

```
$ bash greet.sh Alice
```

Output

```
Hello, Alice!
```

In this case, $1 accesses the first argument passed to the script (Alice).

Using $# (Number of Arguments)

You can use $# to check how many arguments were passed to the script. This is useful for argument validation.

Example:

```
#!/bin/bash
# check_arguments.sh - Check the number of arguments passed

if [ $# -eq 0 ]; then
  echo "No arguments provided!"
else
  echo "You provided $# argument(s)."
Fi
```

If no arguments are provided:

```
$ bash check_arguments.sh
No arguments provided!
```

If arguments are provided:

```
$ bash check_arguments.sh hello world
You provided 2 argument(s).
```

Using $@ and $* (All Arguments)

- **$@**: Expands to all arguments, but each argument is quoted separately.
- **$***: Expands to all arguments as a single string, with each argument separated by a space.

Example:

```
#!/bin/bash
# show_all_arguments.sh - Display all arguments

echo "Arguments using \$@:"
for arg in "$@"; do
  echo "$arg"
done

echo "Arguments using \$*:"
for arg in $*; do
  echo "$arg"
done
```

Running the script:

```
$ bash show_all_arguments.sh one two "three four" five
```

Output

```
Arguments using $@:
one
two
three four
five

Arguments using $*:
one two three four five
```

Notice that $@ preserves the integrity of the argument "three four", treating it as one argument, while $* treats all arguments as separate entities.

Argument Validation Example

You can validate whether the required arguments are passed:

```
#!/bin/bash
# validate_args.sh - Ensure exactly two arguments are passed

if [ $# -ne 2 ]; then
  echo "Usage: $0 <arg1> <arg2>"
  exit 1
fi

echo "Argument 1: $1"
echo "Argument 2: $2"
```

If the user runs the script without arguments or with the wrong number of arguments:

```
$ bash validate_args.sh
Usage: ./validate_args.sh <arg1> <arg2>
```

If the correct number of arguments is provided:

```
$ bash validate_args.sh hello world
Argument 1: hello
Argument 2: world
```