

Command Line Arguments

In the section, we will take a look at "Command Line Arguments" in shell scripts.

We can replace the hardcoded names within our script with the built in variable.

- \$0, \$1, \$2, \$3 ... etc. are built in variables.

Command Line Arguments

```
$ create-and-launch-rocket
$ #modify the create-and-launch-rocket
$ create-and-launch-rocket
$ #modify the create-and-launch-rocket
$ create-and-launch-rocket
$ create-and-launch-rocket/saturn-mission
$0 $1
$ create-and-launch-rocket jupiter-mission
$ create-and-launch-rocket uranus-mission
```

```
create-and-launch-rocket
mission_name=$1
mkdir $mission_name
rocket-add $mission_name
rocket-start-power $mission_name
rocket-internal-power $mission_name
rocket-start-sequence $mission_name
rocket-start-engine $mission_name
rocket-lift-off $mission_name
rocket-status $mission_name
rocket_status=$(rocket-status $mission_name)
echo "Status of launch: $rocket_status"
```

Best Practices

- It is a best practice to assign a variable to a meaningful variable name

```
CODE{CLOUD
```

Command Line Arguments

✓

```
create-and-launch-rocket
mission_name= $1
mkdir $mission_name
rocket-add $mission_name
rocket-start-power $mission_name
rocket-internal-power $mission_name
rocket-start-sequence $mission_name
rocket-start-engine $mission_name
rocket-lift-off $mission_name
rocket-status $mission_name
rocket_status=$(rocket-status $mission_name)
echo "Status of launch: $rocket_status"
```

✗

```
create-and-launch-rocket
mkdir $1
rocket-add $1
rocket-start-power $1
rocket-internal-power $1
rocket-start-sequence $1
rocket-start-engine $1
rocket-lift-off $1
rocket-status $1
rocket_status=$(rocket-status $1)
echo "Status of launch: $rocket_status"
```

Best Practice

"Design your script to be re-usable."

"Script should not require to be edited before running."

"Use command line arguments to pass inputs."