Positional Parameters.

Command line parameters.

Ways to handle User input.

- Scripts interacting with data, variables and files on Linux System
- How to have scripts that interact with the person running the script.
  - ***Command line parameters*** (data values after the command).
  - ***Command line options*** (single-values that modify behaviour of command).
  - Capability to read data directly from the keyboard.

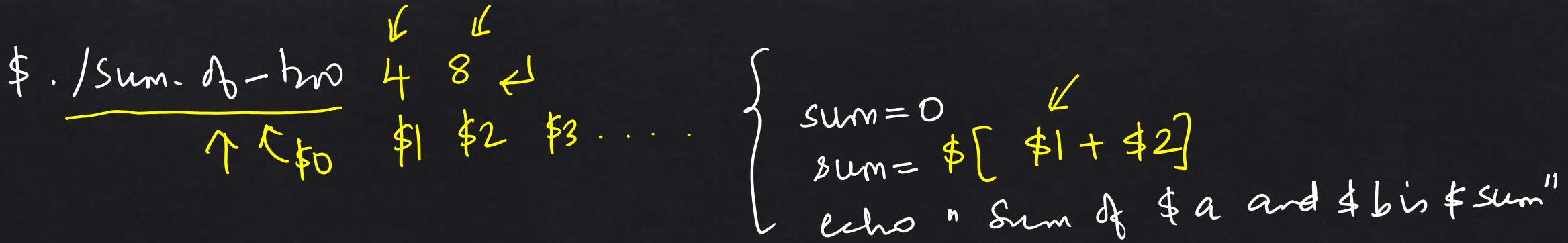
Passing Parameters from Command Line.

- Add data values to the command line when executing the script.

Example: **./sum_of_two 10 40**

$0 $1 — $n

- 10 and 40 are command line parameters to the script sum_of_two



How the script handles input?

- When a shell script is invoked with a set of command line parameters each of these parameters are copied into ***bash shell assigned special variables*** that can be accessed.
- These are called **positional parameters.**
  - **$0:** Variable that contains the name of the script
  - **$1, $2, ….. $n:** 1st, 2nd 3rd command line parameter
  - **$#:** Number of command line parameters
- Standard numbers are used.

Other special positional parameters.

- **$@**: Parameters treated as *separate words*.
- **$*** : Parameters treated as *one word*
- **$?** :Return code ‘exit code’ of the last command

Shifting anyone?

- **Shift command:** Command shifts the positional parameters *by one towards the beginning and drops $1 from the list*. After a shift $2 becomes $1 , and so on.
- Useful command for processing the input parameters one at a time.

Example script

```
1.  #!/bin/bash
2.  # demonstrating the shift command
3.  clear
4.  count=1
5.  echo "Actual number of parameters = $*"
6.  while [ -n "$1" ]
7.  do
8.     echo -e "\tTotal number of parameters = $#"
9.     echo -e "\tparameters = $*"
10.    echo -e "\tParameter#$count = $1"
11.    count=$[ $count + 1 ]
12.    shift
13.    echo -e "\tNow \$1 = $1"
14. done

15. #$./scrt54 1 2 3 54
```