

**REPORT ASSIGNMENT**  
**OPERATING SYTEM A CLASS**



**“Array in Linux Shell Scripting”**

**Name :**

Nurmalita Fitri Ramadani (21083010067)

**Lecturer :**

Mohammad Idhom, S.P., S.Kom., M.T.

Nine Alvariqati (Assistant Lecturer)

**DEPARTMENT OF DATA SCIENCE**  
**COMPUTER SCIENCE FACULTY**  
**UNIVERSITAS PEMBANGUNAN NASIONAL “VETERAN”**  
**JAWA TIMUR**

**2022**

## Array

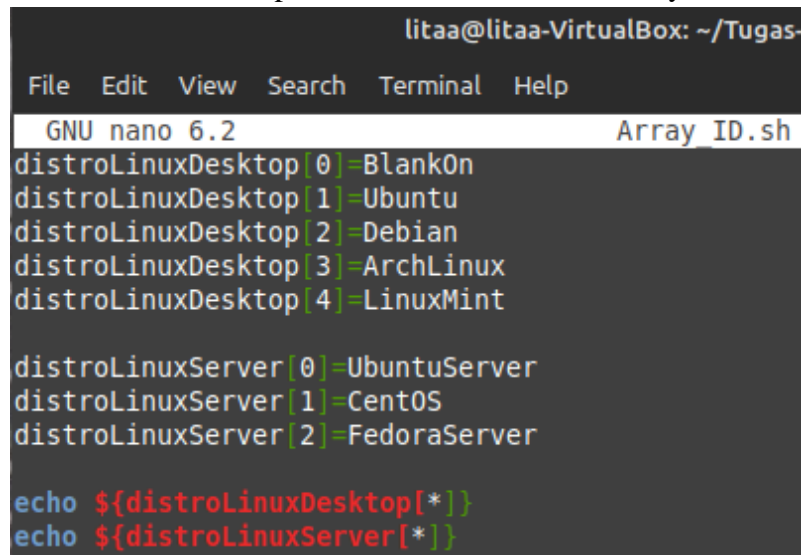
Array is a collection of variables with the same type that stored is a variable with the same name, by assigning an index to variable to differentiate from each variable. There's some type of Array:

- A. Indirect Declaration
- B. Explicit Declaration
- C. Compound Assignment
- D. Multi Dimension

### A. Indirect Declaration

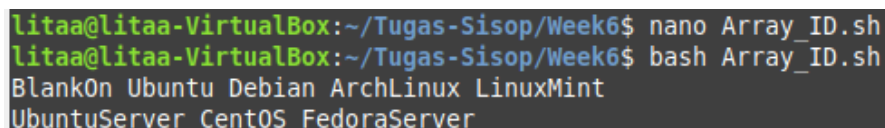
In Array Indirect Declaration, we assign a value in the specified index of the Array Variable. No need to declare it first, the example syntax is: `Array_name[index]=value`.

The picture below shows the written syntax according to the task. `DistroLinuxDesktop` and `DistroLinuxServer` is the `Array_name` and will be the variable of the array. The number inside the `[ ]` stand for the column placement or index of the Array.



```
litaa@litaa-VirtualBox: ~/Tugas-  
File Edit View Search Terminal Help  
GNU nano 6.2 Array_ID.sh  
distroLinuxDesktop[0]=BlankOn  
distroLinuxDesktop[1]=Ubuntu  
distroLinuxDesktop[2]=Debian  
distroLinuxDesktop[3]=ArchLinux  
distroLinuxDesktop[4]=LinuxMint  
  
distroLinuxServer[0]=UbuntuServer  
distroLinuxServer[1]=CentOS  
distroLinuxServer[2]=FedoraServer  
  
echo ${distroLinuxDesktop[*]}  
echo ${distroLinuxServer[*]}
```

Use bash command to see the output of the syntax, and it'll print an output as the picture below shows.



```
litaa@litaa-VirtualBox:~/Tugas-Sisop/Week6$ nano Array_ID.sh  
litaa@litaa-VirtualBox:~/Tugas-Sisop/Week6$ bash Array_ID.sh  
BlankOn Ubuntu Debian ArchLinux LinuxMint  
UbuntuServer CentOS FedoraServer
```

### B. Explicit Declaration

In Explicit Declaration, we declare an array and then assign the value of it. The example syntax is: `declare -a Array_name`. The picture below shows the written syntax according to the task.

```
litaa@litaa-VirtualBox: ~/Tugas
File Edit View Search Terminal Help
GNU nano 6.2 Array_ED.sh
declare -a angka

i=0
while [ $i -le 4 ];
do
    let isi=$i*2;
    angka[$i]=$isi;
    let i=$i+1;
done

echo ${angka[@]}
```

Use bash command to see the output of the syntax, and it'll print an output of Arithmetic numbers that add two to each sequence as the picture below shows.

```
litaa@litaa-VirtualBox:~/Tugas-Sisop/Week6$ nano Array_ED.sh
litaa@litaa-VirtualBox:~/Tugas-Sisop/Week6$ bash Array_ED.sh
0 2 4 6 8
```

### C. Compound Assignment

In Compound Assignment, we declare an array with a set of values, then we can add other values too without having to change the form of the array syntax. The example syntax is: `Array_name=([1]=10 [2]=20 [3]=30)` or `Array_name=(value1 value2...)`. The picture below shows the written syntax according to the task.

```
litaa@litaa-VirtualBox: ~/Tugas-Sisop/Week6
File Edit View Search Terminal Help
GNU nano 6.2 Array_CA.sh *
distroLinuxDesktop=('BlankOn' 'Ubuntu' 'Debian' 'ArchLinux' 'LinuxMint')
distroLinuxServer=('UbuntuServer' 'CentOS' 'FedoraServer')

echo ${distroLinuxDesktop[*]}
echo ${distroLinuxServer[*]}
```

Use bash command to see the output of the syntax, and it'll print an output as the picture below shows.

```
litaa@litaa-VirtualBox:~/Tugas-Sisop/Week6$ nano Array_CA.sh
litaa@litaa-VirtualBox:~/Tugas-Sisop/Week6$ bash Array_CA.sh
BlankOn Ubuntu Debian ArchLinux LinuxMint
UbuntuServer CentOS FedoraServer
```

### D. Multi Dimension

Bash does not have multi-dimensional arrays. Due to Bash provides one dimensional indexed and associative array variables. Any variable can be used as an indexed array, declaring a function will explicitly declare arrays, etc. But it can simulate the effect that is

similar to multi-dimensional associative arrays. The picture below shows the written syntax according to the task.

```
litaa@litaa-VirtualBox: ~/Tugas-Sisop/Week6
File Edit View Search Terminal Help
GNU nano 6.2 Array MD.sh *
array2dimensi="1.1:1.2:1.3:1.4 2.1:2.2:2.3:2.4 3.1:3.2:3.3:3.4"

function dimensiBaris {
    for baris in $array2dimensi
    do
        dimensiKolom `echo $baris | tr : " "`
    done
}

function dimensiKolom {
    for kolom in $*
    do
        echo -n $kolom " "
    done
    echo
}

dimensiBaris
```

Use bash command to see the output of the syntax, and it'll print a 4x3 matrix with values according to the array2dimensional variable. To do a matrix printout, we need 2 function to declare the row dimensions and column dimensions, we can see output as the picture below shows.

```
litaa@litaa-VirtualBox:~/Tugas-Sisop/Week6$ nano Array_MD.sh
litaa@litaa-VirtualBox:~/Tugas-Sisop/Week6$ bash Array_MD.sh
1.1  1.2  1.3  1.4
2.1  2.2  2.3  2.4
3.1  3.2  3.3  3.4
```

### ➤ Practice Task

Buatlah program array yang dapat menghitung nilai IPK mahasiswa yang menerapkan beberapa konsep pemrograman bash seperti diatas dengan ketentuan sbb!

- o user input data arrayIPSMahasiswa[index]
- o  $IPK = (\text{jumlah nilai IPS}) / (\text{jumlah data IPS})$

- First, write the syntax according to the task.

Code:

```
echo "--Simple Program for Counting Student's IPK--"
echo "-----"

declare -a ips_num

echo -n "Enter Total IPS: "
read ips

i=0
while [ $i -lt $ips ];
```

```

do
echo -n "Score Subject( $i ): "
read ips_num[$i];
let i=i+1;
done

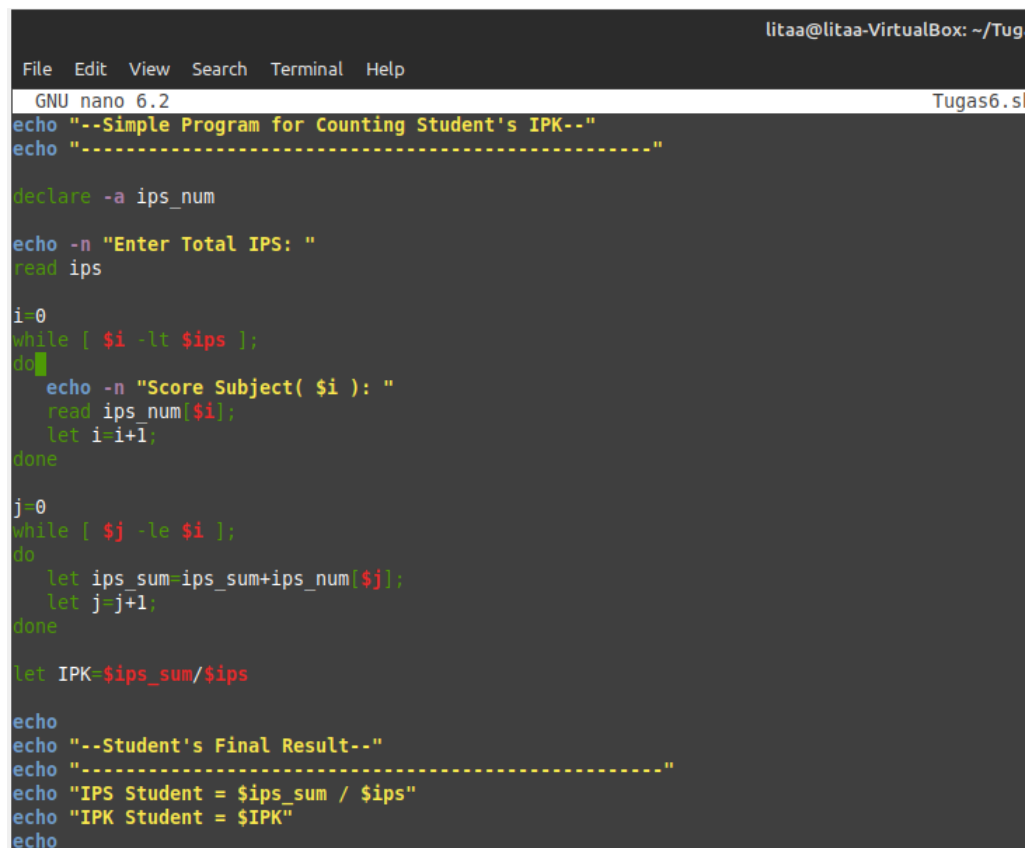
j=0
while [ $j -le $i ];
do
let ips_sum=ips_sum+ips_num[$j];
let j=j+1;
done

let IPK=$ips_sum/$ips

echo
echo "--Student's Final Result--"
echo "-----"
echo "IPS Student = $ips_sum / $ips"
echo "IPK Student = $IPK"
echo

```

In the GNU text editor:



```

litaa@litaa-VirtualBox: ~/Tug
File Edit View Search Terminal Help
GNU nano 6.2 Tugas6.s
echo "--Simple Program for Counting Student's IPK--"
echo "-----"

declare -a ips_num

echo -n "Enter Total IPS: "
read ips

i=0
while [ $i -lt $ips ];
do
echo -n "Score Subject( $i ): "
read ips_num[$i];
let i=i+1;
done

j=0
while [ $j -le $i ];
do
let ips_sum=ips_sum+ips_num[$j];
let j=j+1;
done

let IPK=$ips_sum/$ips

echo
echo "--Student's Final Result--"
echo "-----"
echo "IPS Student = $ips_sum / $ips"
echo "IPK Student = $IPK"
echo

```

- As we done writing the syntax, use bash command to see if the code work and can print out the goals output. As the picture below shows, the code work well as it asking the total ips (index) and we can add data(user input data array) based on the number of index, lastly it'll count the final result (IPK).

```
litaa@litaa-VirtualBox:~/Tugas-Sisop/Week6$ nano Tugas6.sh
litaa@litaa-VirtualBox:~/Tugas-Sisop/Week6$ bash Tugas6.sh
--Simple Program for Counting Student's IPK--
-----
Enter Total IPS: 4
Score Subject( 0 ): 3
Score Subject( 1 ): 4
Score Subject( 2 ): 2
Score Subject( 3 ): 4

--Student's Final Result--
-----
IPS Student = 13 / 4
IPK Student = 3
```