

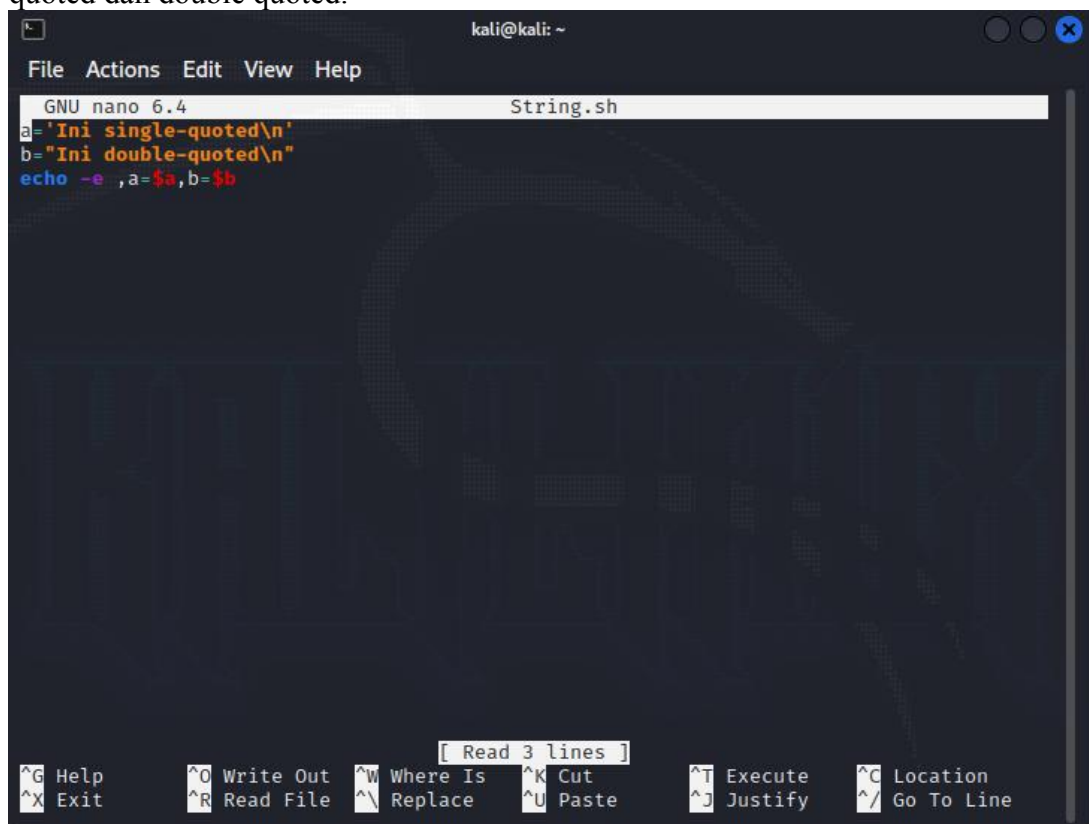
Nama : Andry Syva Maldini  
NPM : 21083010085  
Kelas : Sistem Operasi-A

### Percobaan String (single-quoted) dan (double-quoted):

A quote string atau string yang dikutip adalah sebuah konstanta yang diapit oleh sebuah tanda petik. Tanda petik sendiri memiliki 2 jenis, yakni single-quoted ('...') dan double quote ("...")

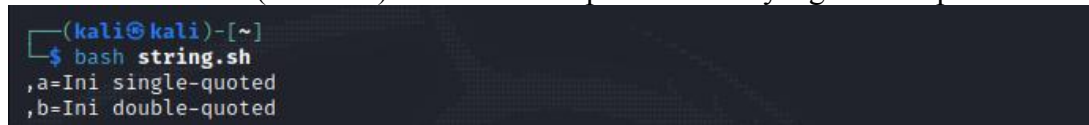
#### a) Nano version

1. Membuat nano file yang diberi nama string.sh
2. Disini saya mencoba membuat sebuah pemrograman mengenai string single quoted dan double quoted.



```
GNU nano 6.4 String.sh
a='Ini single-quoted\n'
b="Ini double-quoted\n"
echo -e ,a=$a,b=$b
```

3. Dapat dilihat pada gambar diatas, bahwa a menggunakan single quoted dan b menggunakan double quoted.
4. \n Pada skrip diatas berfungsi untuk menambahkan “enter” bagi kalimat yang selanjutnya.
5. Echo berfungsi untuk menampilkan teks di layar.
6. Setelah itu save file nano dengan menekan ctrl+s(save) kemudian ctrl+x (exit).
7. Gunakan “bash (namafile)” untuk menampilkan isi file yang kita tampilkan.



```
(kali@kali)-[~]
$ bash string.sh
,a=Ini single-quoted
,b=Ini double-quoted
```

8. Output yang akan dikeluarkan akan terlihat seperti gambar diatas.

b) No nano version (terminal)

1. Tidak jauh dari pada versi file nano dan versi terminal, yang membedakan hanyalah tempatnya.

```
(kali@kali)-[~]
$ a='ini single-quoted\n'

(kali@kali)-[~]
$ b='ini double-quoted\n'

(kali@kali)-[~]
$ echo -e ,a=$a,b=$b
,a=ini single-quoted
,b=ini double-quoted
```

2. Pada skrip diatas, dapat dilihat perbedaan yang signifikan, itu disebabkan adanya pengaruh dari \n yang memiliki “enter” bagi kalimat selanjutnya.

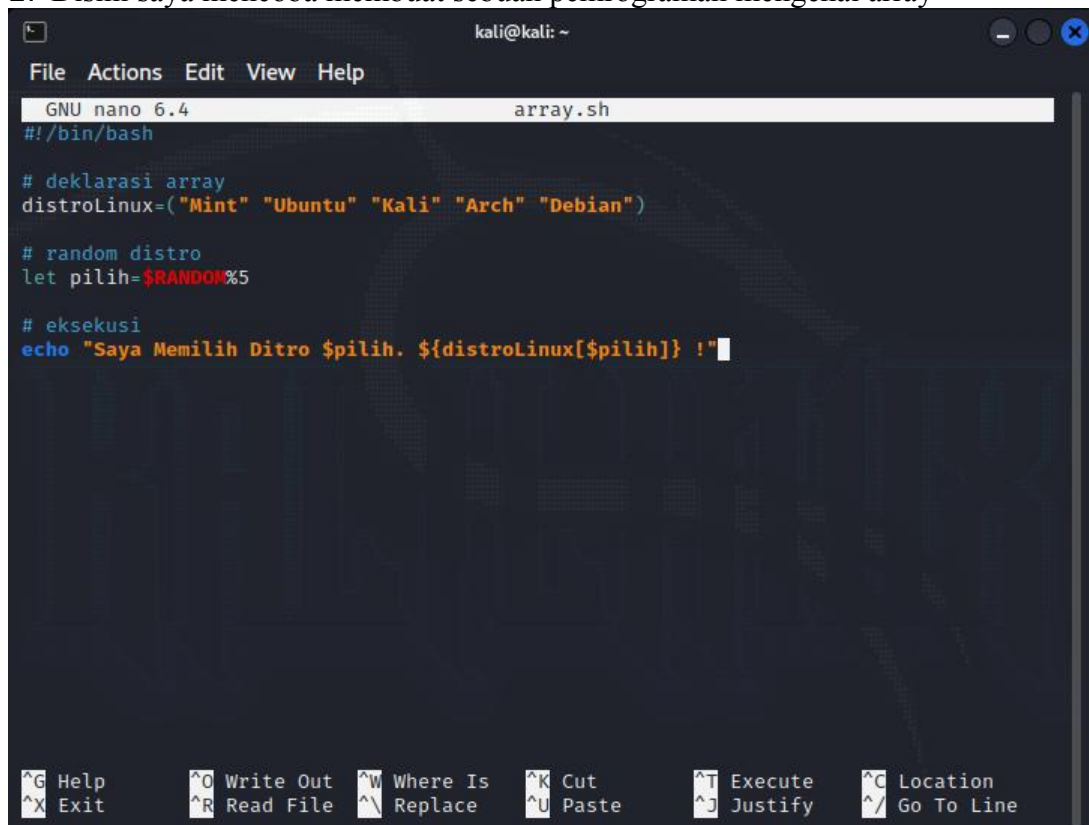
3. Untuk menampilkan teks kita dapat langsung menggunakan “echo”.

### Percobaan Array

Array ialah koleksi dari beberapa data yang mirip/sejenis dengan letak yang berdekatan yang direferensikan dengan index atau lainnya.

1. Membuat nano file yang diberi nama array.sh

2. Disini saya mencoba membuat sebuah pemrograman mengenai array



```
kali@kali: ~
File Actions Edit View Help
GNU nano 6.4 array.sh
#!/bin/bash

# deklarasi array
distroLinux=("Mint" "Ubuntu" "Kali" "Arch" "Debian")

# random distro
let pilih=$RANDOM%5

# eksekusi
echo "Saya Memilih Distro $pilih. ${distroLinux[$pilih]} !"

^G Help      ^O Write Out ^W Where Is  ^K Cut       ^T Execute   ^C Location
^X Exit      ^R Read File ^\ Replace   ^U Paste     ^J Justify   ^_ Go To Line
```

3. Tanda **#!/bin/bash** dalam skrip tes adalah sebuah perintah yang diterjemahkan ke kernel linux untuk mengeksekusi path yang disertakan dalam hal ini program **bash** pada direktori **/bin**. Sebenarnya tanpa tanpa mengikuti baris tersebut anda tetap dapat mengeksekusi skip **bash**, dengan catatan **bash** adalah shell aktif.

4. Seperti yang terlihat pada gambar diatas, pada “distroLinux” saya mendeklarasi sebuah elemen yang berisi dari nama jenis linux, hal tersebut dapat dikatakan sebuah array
5. Di syntax lanjutannya saya memprogram agar output yang dikeluarkan mengambil array secara random dari array yang sudah dibuat.
6. Untuk mengeksekusi sebuah program agar dapat berjalan dan menampilkan hasil menggunakan “echo”.
7. Setelah itu save file nano dengan menekan ctrl+s(save) kemudian ctrl+x (exit).
8. Gunakan “bash (namafile)” untuk memanggil file yang kita tampilkan.

```
(kali㉿kali)-[~]
$ bash array.sh
Saya Memilih Ditro 1. Ubuntu !

(kali㉿kali)-[~]
$ bash array.sh
Saya Memilih Ditro 2. Kali !

(kali㉿kali)-[~]
$ bash array.sh
Saya Memilih Ditro 3. Arch !

(kali㉿kali)-[~]
$ bash array.sh
Saya Memilih Ditro 0. Mint !

(kali㉿kali)-[~]
$ bash array.sh
Saya Memilih Ditro 2. Kali !

(kali㉿kali)-[~]
$ bash array.sh
Saya Memilih Ditro 1. Ubuntu !

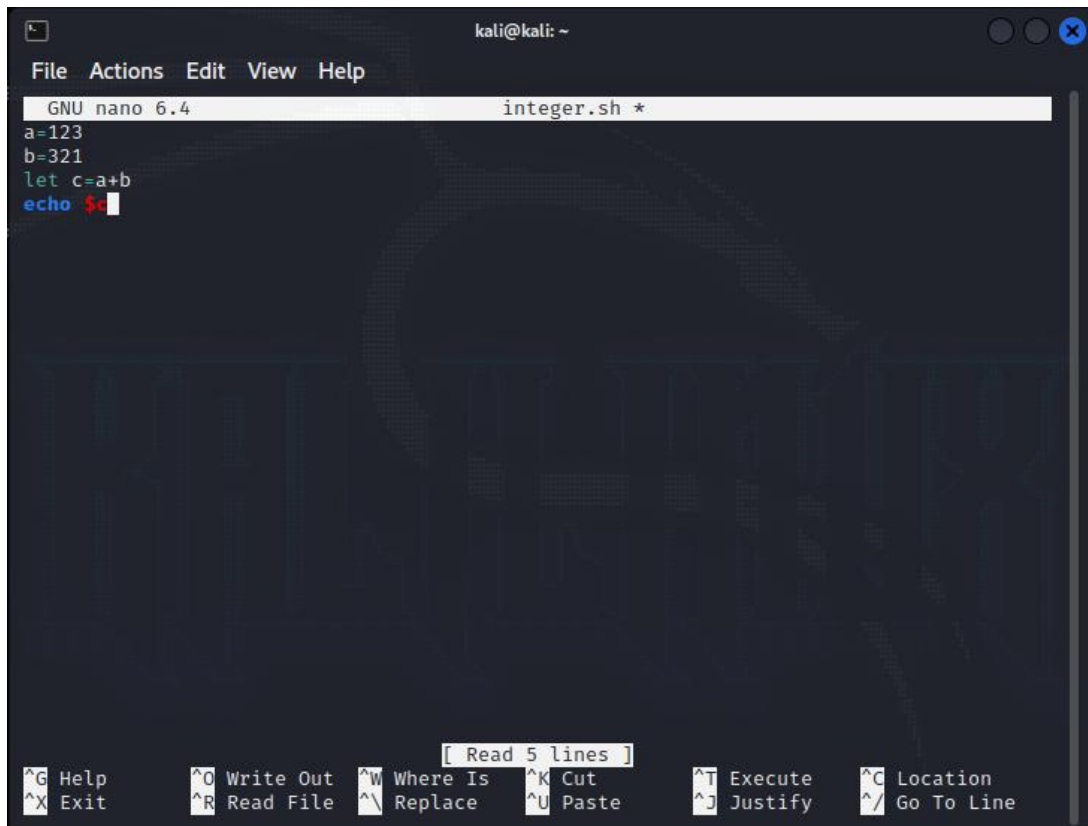
(kali㉿kali)-[~]
$ bash array.sh
Saya Memilih Ditro 3. Arch !
```

9. Hasil atau output yang dikeluarkan akan seperti gambar diatas, program yang dilakukan dapat dikatakan berhasil karena output yang dihasilkan pada setiap pemanggilan memiliki hasil yang berbeda/random.

### Percobaan Integer

Integer sendiri memiliki arti sebuah bilangan bulat, yang dapat terdiri dari bilangan negative, 0, dan bilangan positif. Bilangan pecahan dan decimal tidak termasuk kedalam bilangan integer.

1. Membuat nano file yang diberi nama integer.sh
2. Disini saya membuat sebuah pemrograman mengenai bilangan integer yang dijumlahkan.



```
kali@kali: ~
File Actions Edit View Help
GNU nano 6.4 integer.sh *
a=123
b=321
let c=a+b
echo $c
[ Read 5 lines ]
^G Help ^O Write Out ^W Where Is ^K Cut ^T Execute ^C Location
^X Exit ^R Read File ^_ Replace ^U Paste ^J Justify ^_ Go To Line
```

3. Pada gambar diatas dapat dilihat bahwa a mendeklarasikan angka 123, dan b mendeklarasikan 321.
4. Selanjutnya, c mendeklarasikan penjumlahan antara a dan b.
5. Untuk menghasilkan output pemrograman yang telah dibuat, dapat menggunakan “echo”.
6. Setelah itu save file nano dengan menekan ctrl+s(save) kemudian ctrl+x (exit).
7. Gunakan “bash (namafile)” untuk memanggil file yang kita tampilkan.



```
(kali@kali)-[~]
$ bash integer.sh
444
```

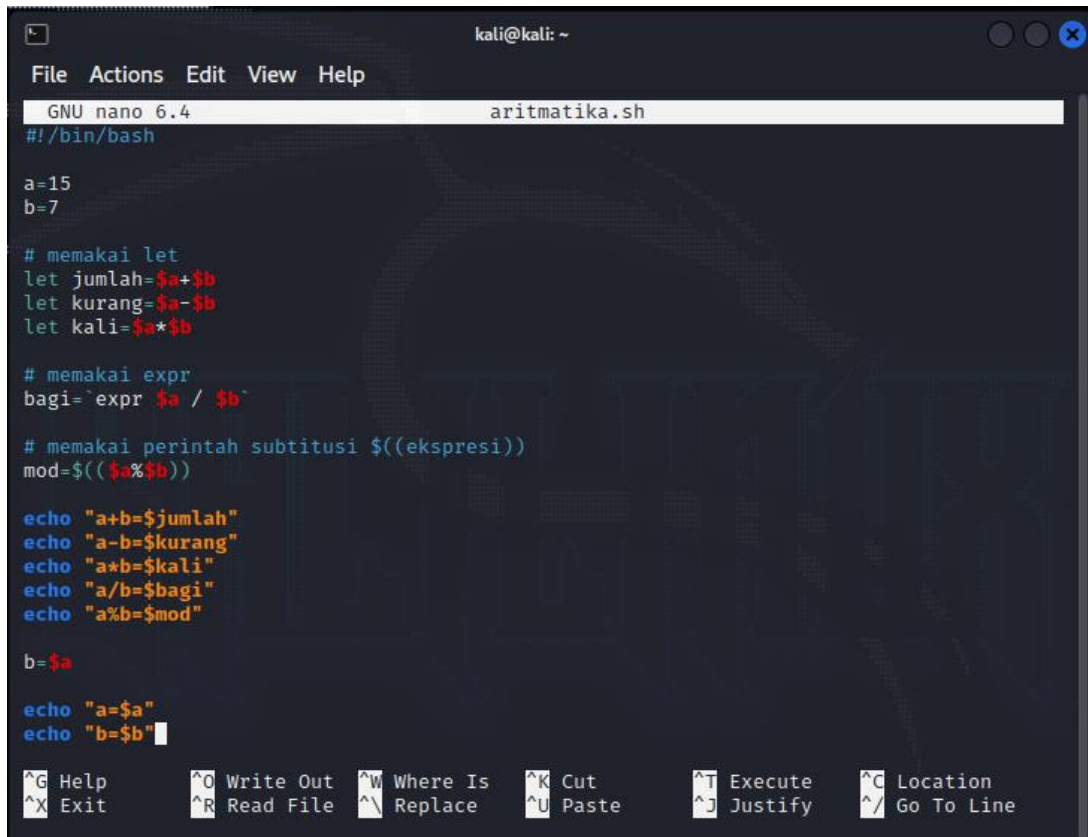
### Percobaan Operasi Aritmatika

Operasi aritmatika adalah operasi dasar dari matematika, yang terdiri dari penjumlahan, pengurangan, perkalian, dan sebagainya. Dalam bash beberapa operasinya dapat dilakukan dengan menggunakan let dan dilanjut dengan pendeklarasian yang ingin dilakukan.

Operasi aritmatika dalam bash terdiri

- + untuk penjumlahan
- - untuk pengurangan
- \* untuk perkalian
- / untuk pembagian
- % untuk modulo atau melihat sisa dari hasil bagi
- = menempatkan nilai di sisi kanan ke variable di sisi kiri
- == membandingkan 2 nilai yang sama
- != membandingkan 2 nilai yang tak sama

1. Membuat nano file yang diberi nama aritmatika.sh
2. Dalam pemrograman kali ini, saya membuat operasi aritmatika yang ada pada bash.



```
kali@kali: ~  
File Actions Edit View Help  
GNU nano 6.4 aritmatika.sh  
#!/bin/bash  
  
a=15  
b=7  
  
# memakai let  
let jumlah=$a+$b  
let kurang=$a-$b  
let kali=$a*$b  
  
# memakai expr  
bagi=`expr $a / $b`  
  
# memakai perintah substitusi $((ekspresi))  
mod=$(( $a % $b ))  
  
echo "a+b=$jumlah"  
echo "a-b=$kurang"  
echo "a*b=$kali"  
echo "a/b=$bagi"  
echo "a%b=$mod"  
  
b=$a  
  
echo "a=$a"  
echo "b=$b"
```

3. Dapat dilihat pada gambar diatas, bahwa a mendeklarasikan angka 15 dan b mendeklarasikan angka 7.
4. Pada pengoperasian aritmatika penjumlahan, pengurangan, dan perkalian kita dapat menggunakan let (build-in) dan untuk pembagian menggunakan expr atau awk (perintah eksternal) dan untuk modulus menggunakan syntax perintah substitusi \$ (ekspresi) seperti diatas.
5. Untuk menampilkan hasil menggunakan "echo"
6. Pada bagian b=\$a dapat dairtikan bahwa deklarasi b yang sebelumnya angka 7, berubah menjadi sama seperti deklarasi a yakni 15.
7. Setelah itu save file nano dengan menekan ctrl+s(save) kemudian ctrl+x (exit).
8. Gunakan "bash (namafile)" untuk memanggil file yang kita tampilkan.

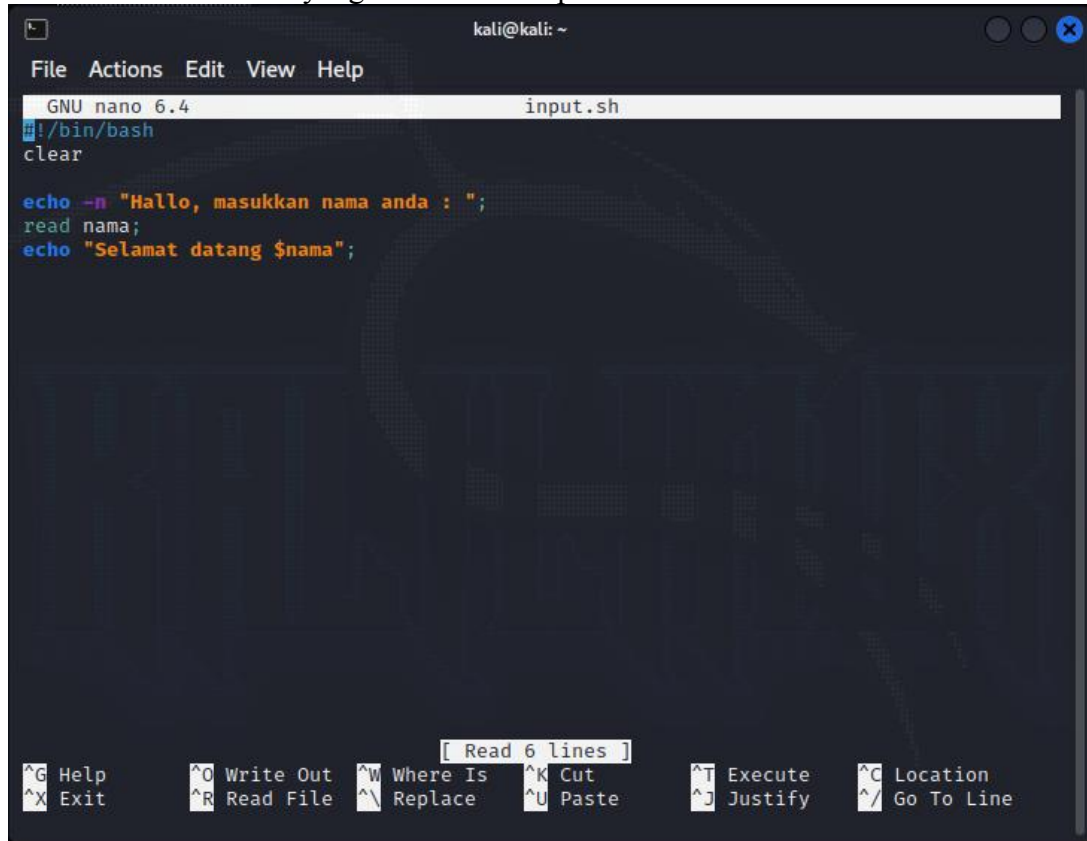


```
(kali@kali)-[~]  
$ bash aritmatika.sh  
a+b=22  
a-b=8  
a*b=105  
a/b=2  
a%b=1  
a=15  
b=15
```

## Percobaan Input

Input memiliki arti data yang dimasukkan oleh user setelah/saat proses output terjadi.

1. Membuat nano file yang diberi nama input.sh



```
kali@kali: ~
GNU nano 6.4 input.sh
#!/bin/bash
clear

echo -n "Hallo, masukkan nama anda : ";
read nama;
echo "Selamat datang $nama";

[ Read 6 lines ]
^G Help      ^O Write Out  ^W Where Is   ^K Cut        ^T Execute    ^C Location
^X Exit      ^R Read File  ^N Replace    ^U Paste      ^J Justify    ^_ Go To Line
```

2. Menggunakan “clear” untuk membersihkan terminal.
3. Read nama yang terdapat diatas memiliki fungsi agar user dapat menginput sebuah nama (string) yang akan di simpan ke dalam variable nama, yang dimana variable tersebut dipanggil dalam echo lanjutannya.
4. Setelah itu save file nano dengan menekan ctrl+s(save) kemudian ctrl+x (exit).
5. Gunakan “bash (namafile)” untuk memanggil file yang kita tampilkan.

```
Hallo, masukkan nama anda : Andry Syva Maldini
Selamat datang Andry Syva Maldini
```

## Percobaan Output

Output sendiri memiliki arti hasil dari suatu proses, baik hasil berupa data maupun informasi yang telah diolah.

- a) Output 1

1. Membuat nano file yang diberi nama output1.sh



```
kali@kali: ~
File Actions Edit View Help
GNU nano 6.4 output1.sh *
#!/bin/bash

matakuliah="Sistem Operasi-A"

echo "Siapa namamu?"
read nama
echo -e "\nHai $nama!\nSelamat datang di pratikum $matakuliah"
```

^G Help    ^O Write Out    ^W Where Is    ^K Cut    ^T Execute    ^C Location  
^X Exit    ^R Read File    ^\_ Replace    ^U Paste    ^J Justify    ^\_/ Go To Line

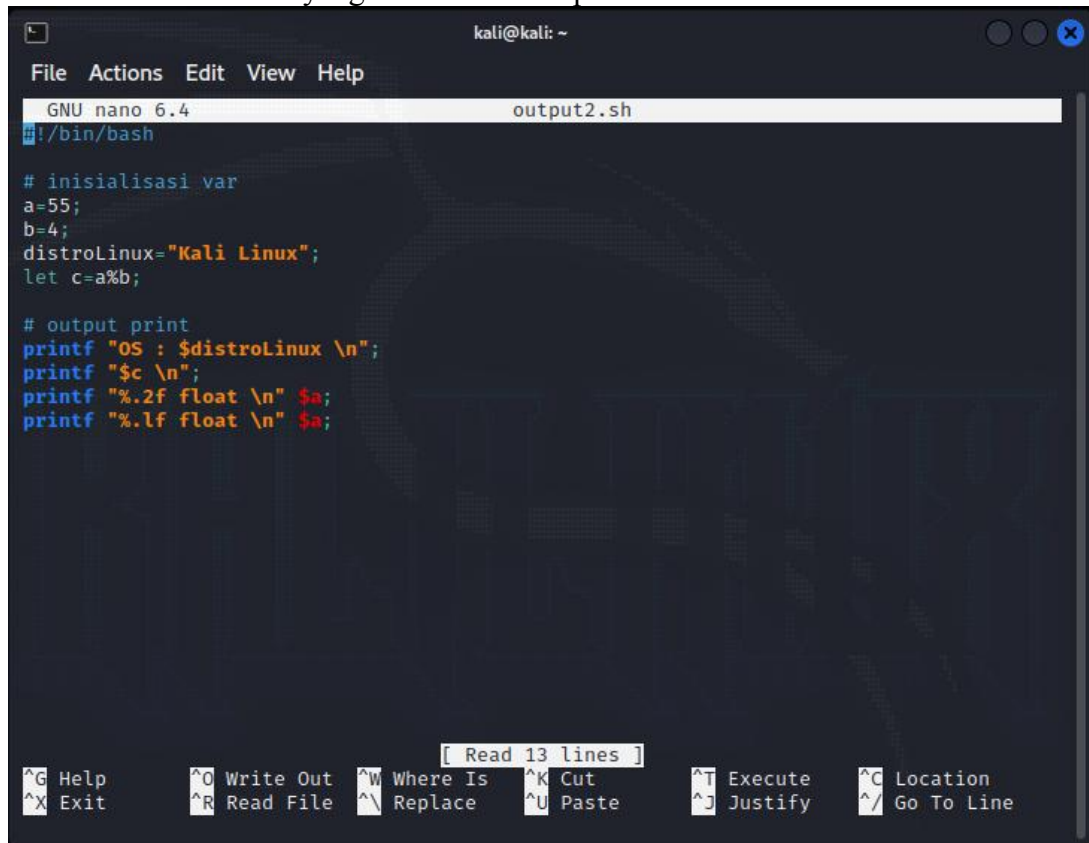
2. Di awal kita dapat melihat bahwa dalam matakuliah mendeklarasikan kalimat Sistem Operasi-A.
3. Read nama yang terdapat diatas memiliki fungsi agar user dapat menginput sebuah nama yang akan disimpan ke dalam variable, yang mana dimana variable tersebut terpanggil dalam echo lajutan.
4. Setelah itu save file nano dengan menekan ctrl+s(save) kemudian ctrl+x (exit).
5. Gunakan “bash (namafile)” untuk memanggil file yang kita tampilkan.

```
(kali@kali)-[~]
$ bash output1.sh
Siapa namamu?
Andry Syva Maldini

Hai Andry Syva Maldini!
Selamat datang di pratikum Sistem Operasi-A
```

b) Output 2

1. Membuat nano file yang diberi nama output2.sh



```
GNU nano 6.4 output2.sh
#!/bin/bash

# inisialisasi var
a=55;
b=4;
distroLinux="Kali Linux";
let c=a%b;

# output print
printf "OS : $distroLinux \n";
printf "$c \n";
printf "%.2f float \n" $a;
printf "%.1f float \n" $a;
```

2. Skrip diatas menunjukan hal pertama yang dilakukan ialah inisialisasi variable.
3. Selanjutnya kita menggunakan operasi aritmatika yaitu modulo.
4. Selain menggunakan “echo” kita juga menggunakan “printf”.
5. %.2 float dapat diartikan bahwa hasil dari pengoperasiannya hanya akan terdapat 2 angka dibelakang koma.
6. Setelah itu save file nano dengan menekan ctrl+s(save) kemudian ctrl+x (exit).
7. Gunakan “bash (namafile)” untuk memanggil file yang kita tampilkan.



```
(kali@kali)-[~]
$ bash output2.sh
OS : Kali Linux
3
55.00 float
55 float
```

## Percobaan Percabangan

Percabangan adalah sebagai kontrol seleksi yang memungkinkan program untuk menjalankan suatu perintah berdasarkan kondisi tertentu.

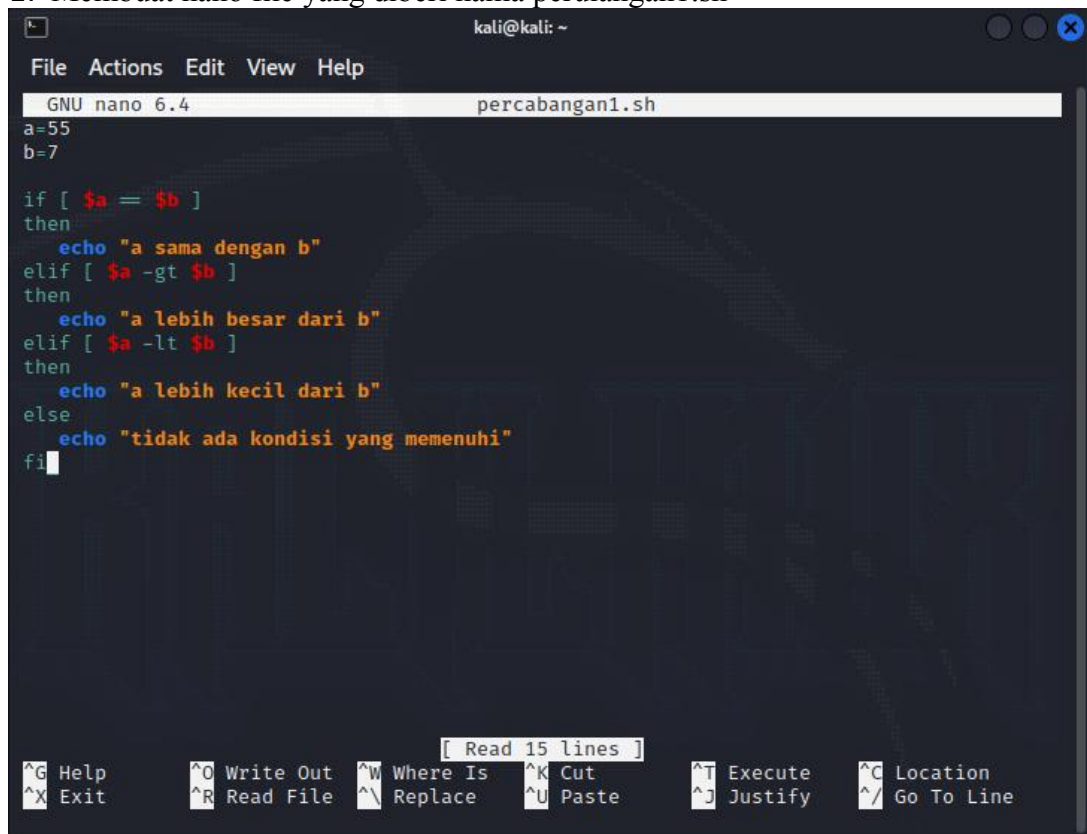
Adapun no operator deskripsi

- -eq nilai kedua operan sama (==)
- -ne nilai kedua operan tidak sama (!=)
- -gt nilai operan kiri lebih besar dari kanan (>)
- -lt nilai operan kanan lebih besar dari kiri (<)
- -ge nilai operan kiri lebih besar sama dengan dari kanan (>=)
- -le nilai operan kanan lebih besar sama dengan dari kiri (<=)



a) Percabangan 1

1. Membuat nano file yang diberi nama perulangan1.sh



```
GNU nano 6.4 percabangan1.sh
a=55
b=7

if [ $a = $b ]
then
    echo "a sama dengan b"
elif [ $a -gt $b ]
then
    echo "a lebih besar dari b"
elif [ $a -lt $b ]
then
    echo "a lebih kecil dari b"
else
    echo "tidak ada kondisi yang memenuhi"
fi
```

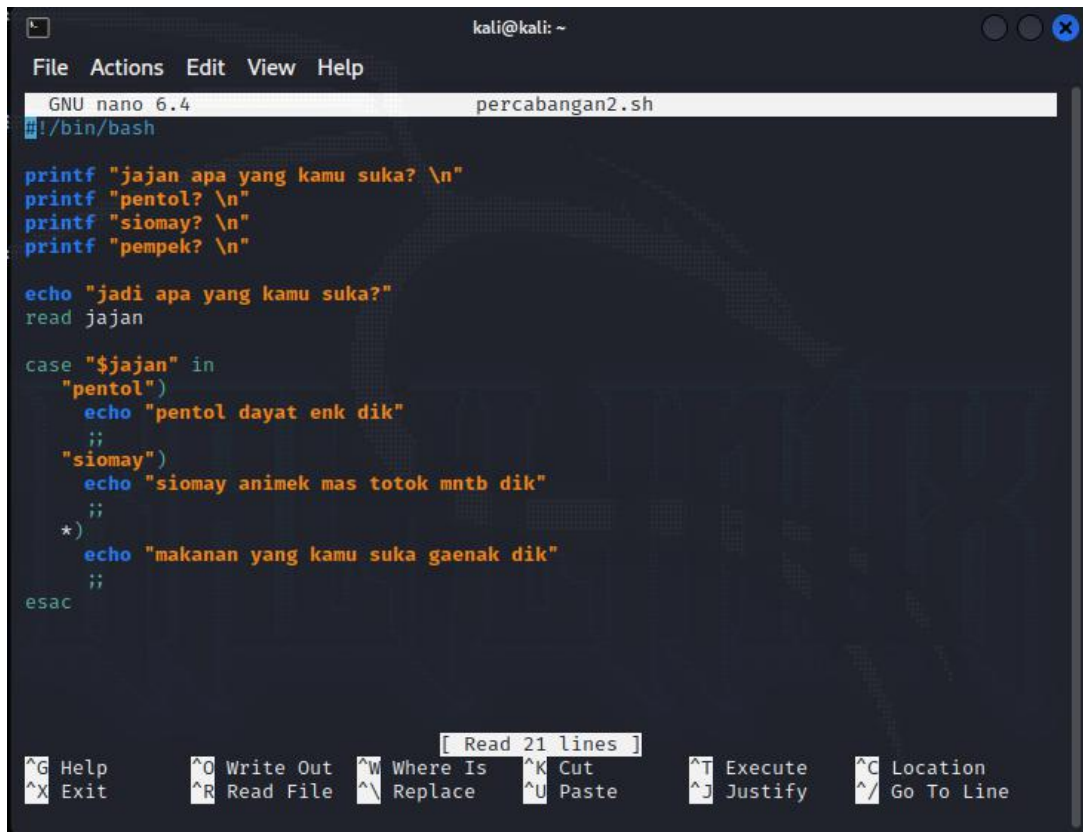
2. Keadaan diatas adalah sebuah kondisi perbandingan angka yang terjadi pada a dan b, dan semua keadaan yang terjadi apabila a dan b berubah nominalnya
3. Setelah itu save file nano dengan menekan ctrl+s(save) kemudian ctrl+x (exit).
4. Gunakan "bash (namafile)" untuk memanggil file yang kita tampilkan.



```
(kali@kali)-[~]
$ bash percabangan1.sh
a lebih besar dari b
```

b) Pecabangan 2

1. Membuat nano file yang diberi nama perulangan2.sh



```
GNU nano 6.4 percabangan2.sh
#!/bin/bash

printf "jajan apa yang kamu suka? \n"
printf "pentol? \n"
printf "siomay? \n"
printf "pempek? \n"

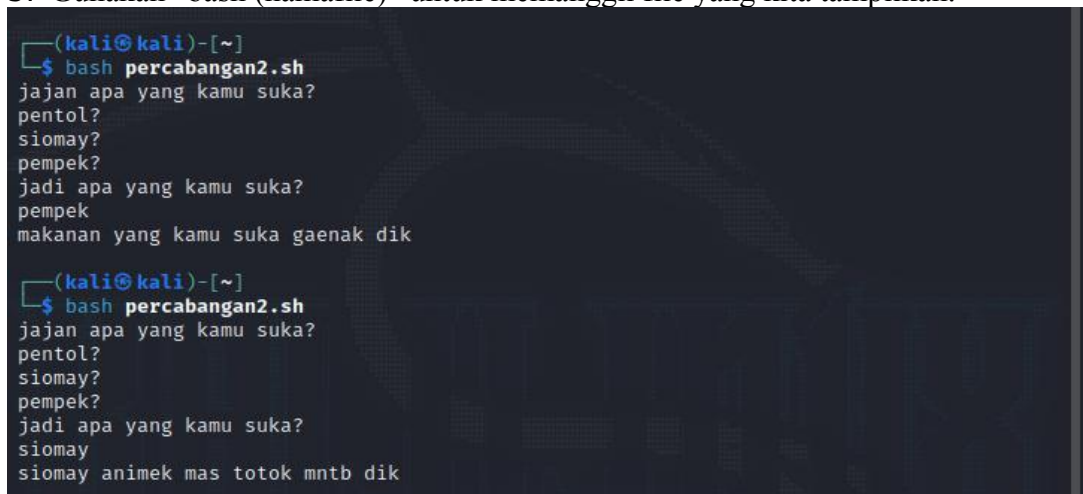
echo "jadi apa yang kamu suka?"
read jajan

case "$jajan" in
    "pentol")
        echo "pentol dayat enk dik"
        ;;
    "siomay")
        echo "siomay animek mas totok mntb dik"
        ;;
    *)
        echo "makanan yang kamu suka gaenak dik"
        ;;
esac
```

[ Read 21 lines ]

^G Help    ^O Write Out    ^W Where Is    ^K Cut    ^T Execute    ^C Location  
^X Exit    ^R Read File    ^\ Replace    ^U Paste    ^J Justify    ^\_ Go To Line

2. Pada pemrograman diatas, dapat dilihat setelah terjadi penawaran mengenai jajan, selanjutnya terdapat sebuah pertanyaan yang dimana nantinya user dapat menginput sebuah jawaban.
3. Dalam case jajan, disana terdapat beberapa kemungkinan user jawaban yang dimana saya telah mempersiapkan balasan dari inputan user yang akan datang.
4. Setelah itu save file nano dengan menekan ctrl+s(save) kemudian ctrl+x (exit).
5. Gunakan "bash (namafile)" untuk memanggil file yang kita tampilkan.



```
(kali@kali)-[~]
$ bash percabangan2.sh
jajan apa yang kamu suka?
pentol?
siomay?
pempek?
jadi apa yang kamu suka?
pempek
makanan yang kamu suka gaenak dik

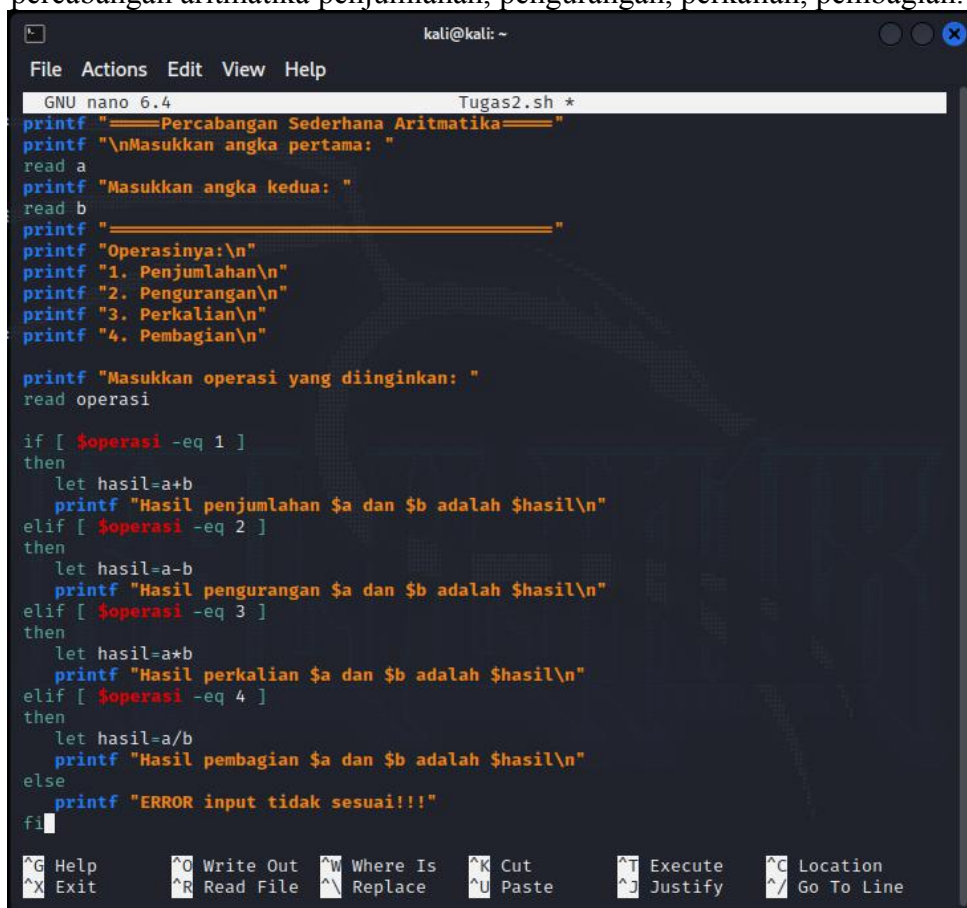
(kali@kali)-[~]
$ bash percabangan2.sh
jajan apa yang kamu suka?
pentol?
siomay?
pempek?
jadi apa yang kamu suka?
siomay
siomay animek mas totok mntb dik
```

6. Dapat dilihat bahwa inputan yang diberi oleh user akan memiliki jawaban yang berbeda pula.

## Tugas

Buatlah pemrograman percabangan sederhana aritmatika yang menerapkan beberapa konsep pemrograman bash seperti diatas!

1. Membuat sebuah file nano, setelahnya diberi nama. Disini saya memberi nama Tugas2.sh
2. Setelah membuat skrip bash sesuai yang diperintah. Disini saya memakai percabangan aritmatika penjumlahan, pengurangan, perkalian, pembagian.



```
GNU nano 6.4 Tugas2.sh *
printf "====Percabangan Sederhana Aritmatika===="
printf "\nMasukkan angka pertama: "
read a
printf "Masukkan angka kedua: "
read b
printf "===="
printf "Operasinya:\n"
printf "1. Penjumlahan\n"
printf "2. Pengurangan\n"
printf "3. Perkalian\n"
printf "4. Pembagian\n"

printf "Masukkan operasi yang diinginkan: "
read operasi

if [ $operasi -eq 1 ]
then
    let hasil=a+b
    printf "Hasil penjumlahan $a dan $b adalah $hasil\n"
elif [ $operasi -eq 2 ]
then
    let hasil=a-b
    printf "Hasil pengurangan $a dan $b adalah $hasil\n"
elif [ $operasi -eq 3 ]
then
    let hasil=a*b
    printf "Hasil perkalian $a dan $b adalah $hasil\n"
elif [ $operasi -eq 4 ]
then
    let hasil=a/b
    printf "Hasil pembagian $a dan $b adalah $hasil\n"
else
    printf "ERROR input tidak sesuai!!!"
fi
```

3. Diharapkan user menginputkan angka pertama dan kedua.
4. Kemudian user memilih ingin melakukan operasi aritmatika apa. untuk menginputkan user hanya perlu menginputkan nomernya saja.
5. Setelah itu save file nano dengan menekan ctrl+s(save) kemudian ctrl+x (exit).
6. Gunakan “bash (namafile)” untuk memanggil file yang kita tampilkan.
7. Hasilnya bisa dilihat gambar dibawah ini.



```
(kali@kali)-[~]
$ bash Tugas2.sh
====Percabangan Sederhana Aritmatika====
Masukkan angka pertama: 100
Masukkan angka kedua: 50
====
Operasinya:
1. Penjumlahan
2. Pengurangan
3. Perkalian
4. Pembagian

Masukkan operasi yang diinginkan: 4
Hasil pembagian 100 dan 50 adalah 2
```