

Perintah Bash Shell pada Linux

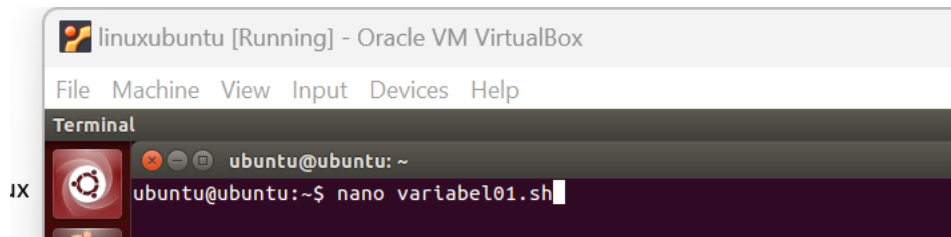
PERINTAH	DESKRIPSI	FORMAT
NANO	Perintah yang digunakan untuk membuat program shell bagi pemula	\$ nano
ECHO	Perintah berfungsi untuk menampilkan variable yang telah dibuat	\$ echo
READ	Perintah ini digunakan untuk memberi nama dan isi dari suatu variabel	\$ read
IF	Perintah ini digunakan untuk membandingkan exit status pada program shell	\$ if

Shell adalah Command executive, artinya program yang menunggu intruksi dari pemakai, memeriksa syntax dari instruksi yang diberikan, kemudian mengeksekusi perintah tersebut.

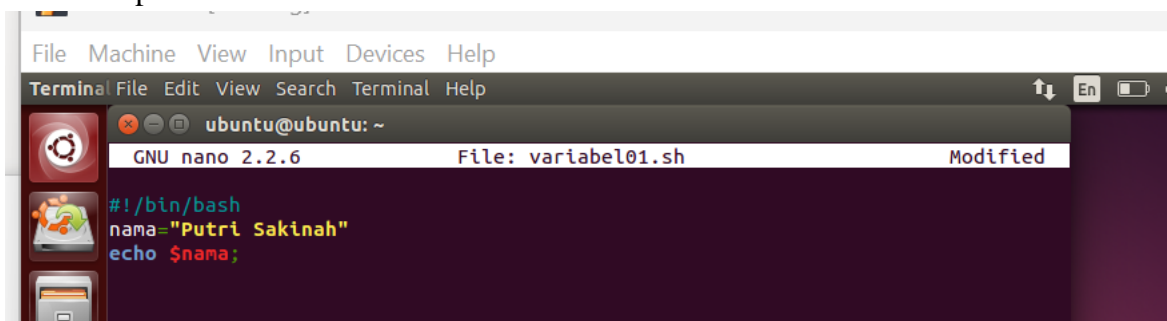
- /bin/sh : Bourno shell, dirancang oleh Steve Bourne dari AT&T
- /bin/csh : Dikembangkan oleh UNIX Beerkeley yang dikenal dengan C – Shell
- /bin/bash : Kompatibel dengan Bourne Shell dan juga mengadaptasi kemampuan KomShell.

1. Instalasi nano di terminal linux

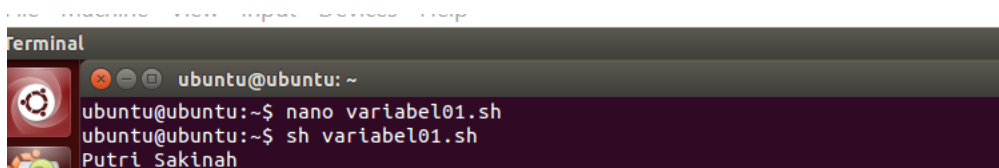
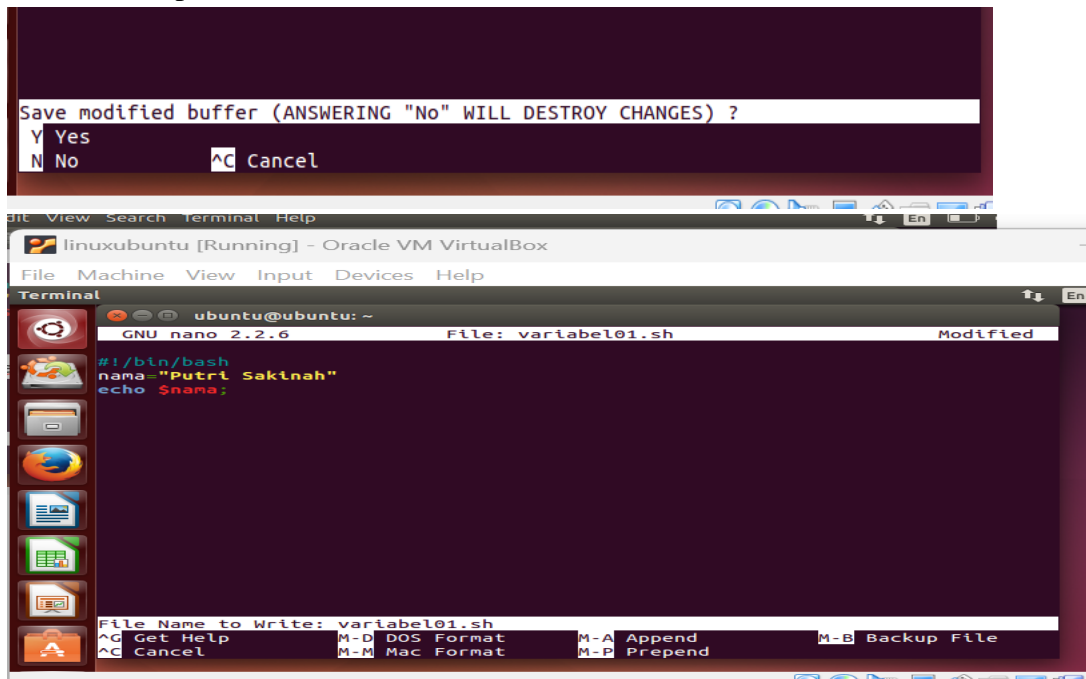
```
yum install nano
```



2. Membuat perintah di nano shell

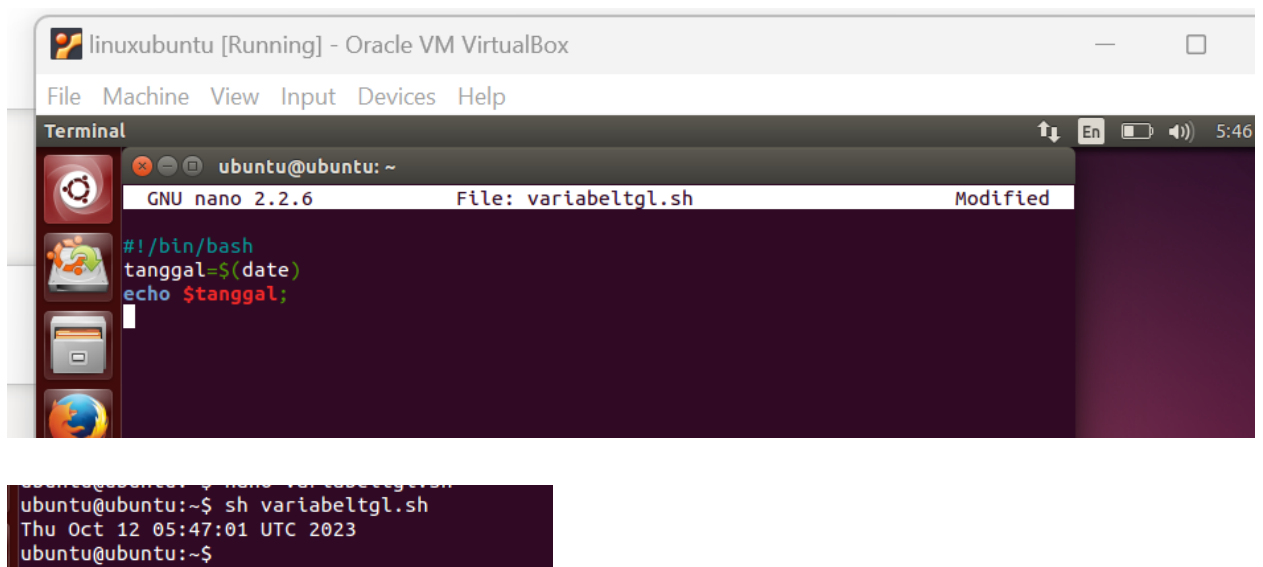


3. Lalu ketikkan perintah ctrl+x



Lalu tekan Enter

4. Membuat pemanggilan tanggal pada nano bash shell



5. Perintah data alamat melalui nano shelln Buat prog2.sh dengan perintah nano **prog2.sh**

```
GNU nano 2.2.6 File: prog2.sh
#!/bin/sh
#prog2.sh
#membaca nama dan alamat

echo -n "Nama anda : "
read nama
echo -n "Alamat : "
read alamat
echo -n "Kota : "
read kota

echo
echo "Hasil adalah : $nama, $alamat di $kota"
```

```
ubuntu@ubuntu:~$ nano prog2.sh
ubuntu@ubuntu:~$ sh prog2.sh
Nama anda : Putri
Alamat : Sakinah
Kota : Padang

Hasil adalah : Putri, Sakinah di Padang
ubuntu@ubuntu:~$
```

6. Operator logical if else pada nano bash shell

- Buatlah file program prog7.sh

```
#!/bin/sh
# prog07.sh
echo -n "INCOME = "
read INCOME
if [ $INCOME -ge 0 -a $INCOME -le 10000 ]
then
    BIAYA=10
elif [ $INCOME -gt 10000 -a $INCOME -le 25000 ]
then
    BIAYA=25
else
    BIAYA=35
fi
echo "BIAYA = $BIAYA"
```

- Jalankan file prog7.sh dan masukkan income untuk INCOME=500,20000,28000

```
$ . prog07.sh [INCOME=5000]
INCOME = 5000
BIAYA = 10
$ . prog07.sh [INCOME=20000]
INCOME = 20000
BIAYA = 25
$ . prog07.sh [INCOME=28000]
INCOME = 28000
BIAYA = 35
```

Analisa : Pada percobaan ini, kita memasukkan angka kedalam variable INCOME. Lalu, dicek dengan kondisional if. `-ge` adalah operator lebih dari sama dengan, `-a` adalah operator AND, dan `-le` adalah operator kurang dari sama dengan. Jadi, kondisional pertama mengecek apakah nilai income berada pada range 0-10000. Jika iya, maka nilai dari variable biaya adalah 10. Jika tidak dicek lagi pada kondisional kedua. `-gt` adalah operator lebih besar dari. Jadi, kondisional kedua adalah mengecek apakah nilai income berada pada range

antara 10000-25000, jika iya nilai BIAYA menjadi 25. Apabila semua kondisional tidak memenuhi nilai biaya akan dijadikan 35. Pada akhir program akan dieksekusi mencetak nilai biaya

7. Hitungan Aritmetika

- Menggunakan utilitas expr

```
$ expr 5 + 1
6
$ A=5
$ expr $A + 2
7
$ expr $A - 4
1
$ expr $A * 2
expr: syntax error
$ expr $A \* 2
10
$ expr $A / 6 + 10
10
$ expr 17 % 5
2
```

Expr adalah utilitas untuk melakukan aritmetika sederhana yaitu tambah, kurang, dan modulus dan langsung mencetak hasilnya secara langsung. Apabila selain itu, operatornya harus ditambah dengan backslash

8. Melihat pengguna aktif dan tanggal

Buatlah perintah `$nano prog03.sh` . lalu ketikkan perintah untuk memanggil nama pengguna seperti dibawah ini. Setelah selesai lalu exit dan save

```
GNU nano 2.9.3 prog03.sh

#!/bin/sh
# prog03.sh
#
NAMA=`whoami`

echo Nama Pengguna Aktif adalah $NAMA

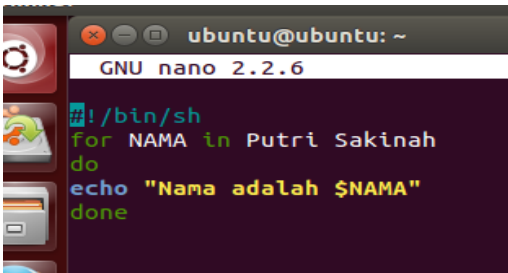
tanggal=`date | cut -c1-10`

echo Hari ini tanggal $tanggal
```

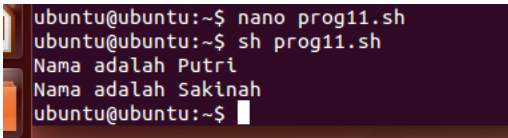
Ketika dijalankan akan muncul seperti ini

```
Nama Pengguna Aktif adalah user
Hari ini tanggal Sab Apr 25
```

9. Konstruksi for-do-done



```
ubuntu@ubuntu: ~  
GNU nano 2.2.6  
#!/bin/sh  
for NAMA in Putri Sakinah  
do  
echo "Nama adalah $NAMA"  
done
```



```
ubuntu@ubuntu:~$ nano prog11.sh  
ubuntu@ubuntu:~$ sh prog11.sh  
Nama adalah Putri  
Nama adalah Sakinah  
ubuntu@ubuntu:~$
```

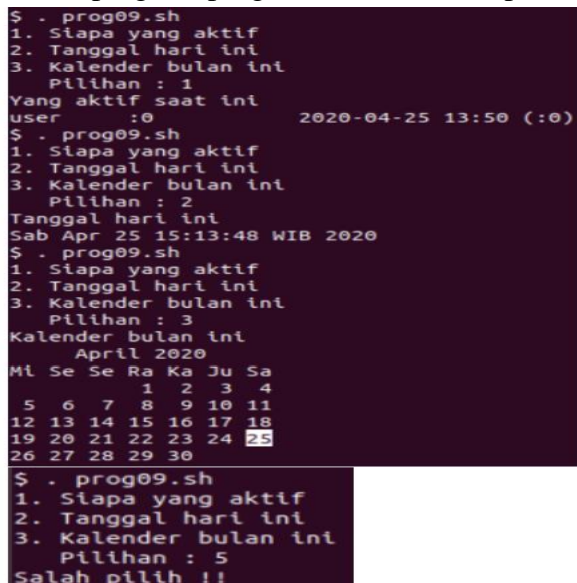
10. Konstruksi case – esac

Buatlah file prog9.sh



```
#!/bin/sh  
echo "1. siapa yang aktif"  
echo "2. tanggal hari ini"  
echo "3. kalender bulan ini"  
echo -n "Pilihan ; "  
read PILIH  
case $PILIH in  
1)  
    echo "yang aktif saat ini"  
    who  
    ;;  
2)  
    echo "tanggal haari ini"  
    date  
    ;;  
3)  
    eecho "kalenderr bulan ini"  
    cal  
    ;;  
*)  
    echo "saalah pilih !"  
    ;;
```

Jalankan program prog09.sh. coba beberapa kali dengan inputan berbeda



```
$ . prog09.sh  
1. Siapa yang aktif  
2. Tanggal hari ini  
3. Kalender bulan ini  
Pilihan : 1  
Yang aktif saat ini  
user      :0          2020-04-25 13:50 (:0)  
$ . prog09.sh  
1. Siapa yang aktif  
2. Tanggal hari ini  
3. Kalender bulan ini  
Pilihan : 2  
Tanggal hari ini  
Sab Apr 25 15:13:48 WIB 2020  
$ . prog09.sh  
1. Siapa yang aktif  
2. Tanggal hari ini  
3. Kalender bulan ini  
Pilihan : 3  
Kalender bulan ini  
April 2020  
Mo Se Sa Ra Ka Ju Sa  
    1  2  3  4  
5  6  7  8  9 10 11  
12 13 14 15 16 17 18  
19 20 21 22 23 24 25  
26 27 28 29 30  
$ . prog09.sh  
1. Siapa yang aktif  
2. Tanggal hari ini  
3. Kalender bulan ini  
Pilihan : 5  
Salah pilih !!
```

11. Pada percobaan ini, variable F diisi dengan string nama semua file yang ada pada direktori home lalu dicetak satu persatu secara berulang dengan looping for

```
#!/bin/sh
# Prog: prog12.sh

for F in *
do
    echo $F
done
```

Jalankan program prog12.sh

```
$ . prog12.sh
bin
Desktop
Documents
Downloads
examples.desktop
Music
Pictures
prog06.sh
prog07.sh
prog08.sh
prog09.sh
prog10.sh
prog11.sh
prog12.sh
prog1.sh
Public
Templates
Videos
```

12. Konstruksi while-do-done

Buat file prog13.sh

```
#!/bin/sh
# Prog: prog13.sh

PILIH=1
while [ $PILIH -ne 4 ]
do
    echo "1. Siapa yang aktif"
    echo "2. Tanggal hari ini"
    echo "3. Kalender bulan ini"
    echo "4. Keluar"
    echo "Pilihan : \c"
    read PILIH
    if [ $PILIH -eq 4 ]
    then
        break
    fi
    clear
done
echo "Program berlanjut di sini setelah break"
```

Jalankan program prog13.sh

```
$ . prog13.sh
1. Siapa yang aktif
2. Tanggal hari ini
3. Kalender bulan ini
4. Keluar
Pilihan : \c
4
Program berlanjut di sini setelah break
```

Program ini menggunakan looping while dengan pengkondisian apabila variable PILIH tidak sama dengan (-ne) 4, program akan terus diulang sampai user memilih angka 4 yang artinya looping berhenti. Perintah clear digunakan untuk membersihkan/mengosongkan baris.

13. Dalam input nilai kurang dari 100 akan mencetak string , dan apabila input nilai lebih dari 100 akan mengeksekusi instruksi dummy. Sehingga tidak akan melakukan apaapa dan exit status bernilai TRUE

Buat file prog14.sh

```
#!/bin/sh
# Prog: prog14.sh

echo -n "Masukkan nilai : "
read A
if [ $A -gt 100 ]
then
    :
else
    echo "OK !"
fi
```

Jalankan program prog14.sh dengan input berbeda

```
$ . prog14.sh
Masukkan nilai : 90
OK !
$ . prog14.sh
Masukkan nilai : 200
$
```

14. Fungsi

Buat file fungsi.sh

```
#!/bin/sh
# Prog: fungsi.sh

F1( ) {
    echo "Fungsi F1"
    return 1
}
echo "Menggunakan Fungsi"
F1
F1
echo $?
```

Jalankan program fungsi.sh

```
$ . fungsi.sh
Menggunakan Fungsi
Fungsi F1
Fungsi F1
1
```

Untuk menjalankan fungsi kita bisa memanggil dengan perintah nama fungsinya langsung

15. Menggunakan variable pada fungsi dengan memodifikasi file fungsi.sh

```
#!/bin/sh
# Prog: fungsi.sh

F1( )
{
    Honor=10000
    echo "Fungsi F1"
    return 1
}
echo "Menggunakan Fungsi"
F1
F1
echo "Nilai balik adalah $?"
echo "Honor = $Honor"
```

Jalankan program fungsi.sh

```
$ . fungsi.sh
Menggunakan Fungsi
Fungsi F1
Fungsi F1
Nilai balik adalah 1
Honor = 10000
```

16. File .bash_logout akan dieksekusi sesaat sebelum logout, berfungsi sebagai house clearing jobs, artinya membersihkan semuanya, misalnya menghapus temporary file atau job lainnya. Melihat file .bash_logout dengan instruksi

```
$ cat .bash_logout
# ~/.bash_logout: executed by bash(1) when login shell exits.

# when leaving the console clear the screen to increase privacy

if [ "$SHLVL" = 1 ]; then
    [ -x /usr/bin/clear_console ] && /usr/bin/clear_console -q
fi
```

Perintah ini digunakan untuk eksekusi sebelum logout dan menghapus temporary file dan job.

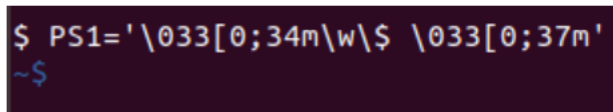
17. Mengubah Prompt Shell

Prompt Bash shell dikonfigurasi dengan men-setting nilai variabel PS1. Selain menampilkan string statik sebagai prompt, Anda dapat menampilkan menjadi dinamis. Contohnya, apabila ingin menunjukkan current directory atau current time. Ketik PS1='\t:' dan tekan Enter untuk menampilkan waktu sistem dalam format 24 jam sebagai prompt Bash. Format dalam HH:MM:SS
\$ PS1='\t:'

```
$ PS1='\t:'
11:37:37
```

Perintah ini untuk menampilkan waktu dalam format HH:MM:SS

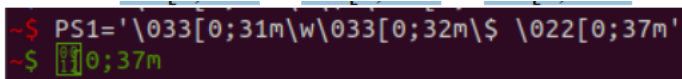
18. Prompt BASH dapat ditampilkan berwarna dengan melakukan setting colorsetting string. Sebagai contoh, prompt BASH di-set dengan \w\$, akan menampilkan current working directory yang diikuti \$ (atau # jika anda login sebagai root). Untuk setting warna menjadi biru ketikkan berikut :
- ```
$ PS1='\033[0;34m\w\$ \033[0;37m'
```



```
$ PS1='\033[0;34m\w\$ \033[0;37m'
```

Perintah ini digunakan untuk mengubah warna prompt bash menjadi biru.

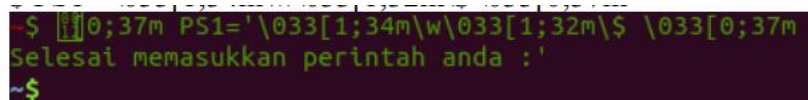
19. Bila menginginkan beberapa warna, ketikkan perintah berikut :
- ```
$ PS1='\033[0;31m\w\033[0;32m\$ \033[0;37m'
```



```
~$ PS1='\033[0;31m\w\033[0;32m\$ \033[0;37m'
```

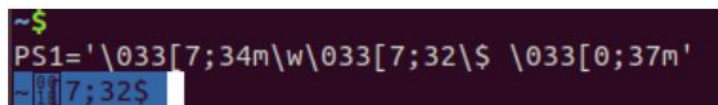
Perintah ini digunakan untuk mengubah warna prompt bash menjadi beberapa warna

20. Anda bisa menampilkan atribut visual seperti lebih terang, berkedip dan warna kebalikannya. Untuk menampilkan prompt yang lebih terang, atribut control diganti 1, seperti perintah berikut :
- ```
$ PS1='\033[1;34m\w\033[1;32m\$ \033[0;37m'
```



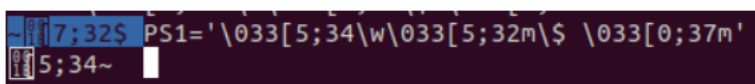
```
~$ PS1='\033[1;34m\w\033[1;32m\$ \033[0;37m'
```

21. Untuk menampilkan prompt dengan warna berkebalikan, atribut control diganti 7, seperti perintah berikut :
- ```
$ PS1='\033[7;34m\w\033[7;32m\$ \033[0;37m'
```



```
~$ PS1='\033[7;34m\w\033[7;32m\$ \033[0;37m'
```

22. Untuk menampilkan prompt berkedip, atribut control diganti 5, seperti perintah berikut :
- ```
$ PS1='\033[5;34m\w\033[5;32m\$ \033[0;37m'
```



```
~$ PS1='\033[5;34m\w\033[5;32m\$ \033[0;37m'
```

Perintah ini untuk mengubah tampilan prompt bash menjadi berkedip

## 23. Percobaan Job Control

### Proses foreground

```
mahiraardelia@ubuntu:~$ ps x
PID TTY STAT TIME COMMAND
 913 ? Ss 0:00 /lib/systemd/systemd --user
 914 ? S 0:00 (sd-pam)
 928 tty1 S 0:00 -bash
1114 tty1 R+ 0:00 ps x
```

Analisis :

Menampilkan proses pada foreground

### Proses background

```
mahiraardelia@ubuntu:~$ ps c > hasil &
[1] 1116
```

Analisa :

Menampilkan jumlah proses yang berjalan

Setiap job mempunyai PID yang tunggal (unique). Untuk melihat jobs yang aktif

```
mahiraardelia@ubuntu:~$ jobs
[1]+ Done ps c > hasil
```

Analisa :

Melihat jobs yang aktif