

LAPORAN PRAKTIKUM SISTEM OPERASI

Praktikum 6a

Oleh:
FARID AZIZ WICAKSONO
NIM. 1841720094



JURUSAN TEKNOLOGI INFORMASI
PROGRAM STUDI TEKNIK INFORMATIKA
POLITEKNIK NEGERI MALANG
MARET 2019

1. SHELL SCRIPT

Shell script dibuat dengan editor teks (ASCII editor) dan umumnya diberikan ekstensi **“.sh”**. shell script selalu diawali dengan komentar, yang dimulai dengan tanda **#**, disambung dengan **!** dan nama shell yang digunakan. Contoh:

```
#!/bin/sh           (1)
# Program shell    (2)
#
Var1=x             (3)
Var2=8
```

- (1) Awal dari program shell, komentar awal ini akan dibaca oleh system, kemudian system mengaktifkan program shell (/bin/sh) yang tertera ditempat tersebut. Program shell dapat dipilih, misalnya /bin/csh, /bin/ksh dan lainnya.
- (2) Adalah komentar, sebagai dokumentasi, baris ini akan diabaikan oleh program shell.
- (3) Penggunaan variable (assignment), tidak boleh ada spasi diantara nama variable dan konstanta.

2. VARIABEL

Dalam memproses data di komputer, kita menyimpan data tersebut dalam memori komputer. Memori tersebut dibagi dalam lokasi-lokasi dengan alamat tertentu. Programmer harus memberi nama lokasi tersebut dengan nama variabel.

Ada 2 macam variabel dalam Linux, variabel sistem biasanya dituliskan dengan huruf besar semua dan variabel user. Cara mendeklarasikan variabel juga hampir sama: **variablename=value** Harap diperhatikan, tidak boleh ada spasi!

Contoh: **\$ x=10**

artinya mengisi angka 10 ke dalam variabel **x** Contoh variabel sistem:

Variabel sistem	Artinya
BASH=/bin/bash	Nama shell
COLUMNS=80	No. kolom pada layar
HOME=/home/yanti	Direktori home
LINES=25	No. baris pada layar
OSTYPE=Linux	Nama sistem operasi
SHELL=/bin/bash	Nama shell
USERNAME=yanti	Nama orang yang sedang bekerja

Variabel User

Aturan penamaan variabel memiliki aturan yang hampir sama dengan penamaan dalam pemrograman komputer bahasa lain:

- Nama variabel harus dimulai dengan huruf atau underscore (_), diikuti dengan huruf lain atau angka;
- Jangan menggunakan spasi, seperti sudah dicontohkan di atas;
- Variabel adalah case-sensitive, huruf besar dan huruf kecil berbeda; □ Variabel kosong (NULL) tidak akan menghasilkan nilai;
- Jangan menggunakan karakter khusus seperti: ?, *, dll untuk nama variabel.

3. MEMBACA KEYBOARD

Nilai variable dapat diisi melalui keyboard (stdin) dengan intruksi read. Program yang selama ini kita buat, nilainya telah ditentukan sebelumnya, program- program tersebut tidak mendapatkan input dari user. Program - program tersebut hanya melakukan hal yang sama setiap waktu.

4. PARAMETER

Sebuah program shell dapat mempunyai parameter sebanyak 9 buah dan direpresentasikan melalui variable khusus yaitu variable \$!, \$2, \$3, \$4, \$5, \$6, \$7, \$8, \$9. Nama program shell || (nama script) direpresentasikan melalui variable \$0. Jumlah parameter dinyatakan sebagai \$#. Bila tidak memberikan parameter maka nilai \$# adalah 0. Shell variable \$* menyatakan seluruh string yang menjadi parameter / argument sebuah script (\$@ mempunyai arti sama). \$\$ menyatakan nomor proses id/pid dari script yang dijalankan. Pid ini akan terus berubah (umumnya) menaik, setiap kali proses berjalan.

5. STATUS EXIT

Setiap program setelah selesai dieksekusi akan memberikan informasi melalui variable special \$?. Indikasi yang diberikan adalah :

- Bila program berakhir dengan sukses, \$?=0
- Bila program berakhir dengan error, \$?tidak samadengan 0. nilai dari status exit dapat dilihat melalui instruksi **echo \$?**

6. KONSTRUKSI IF

```
if instruksi-awal then
    instruksi1
    instruksi2
    .....
fi
```

If akan mengeksekusi instruksi awal, dan exit status dari instruksi tersebut akan menjadi kondisi. Bila 0, maka instruksi selanjutnya masuk ke dalam blok then. Bila tidak 0, maka alur program diteruskan setelah kata kunci **fi**.

7. KONSTRUKSI IF THEN ELSE

Statement builtin if berfungsi untuk melakukan seleksi berdasarkan suatu kondisi tertentu.

Secara umum ada dua bentuk umum sintak perintah if , seperti ditunjukkan dibawah ini

```
if instruksi1 then
    instruksi1.1    instruksi1.2
    ..... else    instruksi2.1
    instruksi2.2
    .....
fi
```

Kalau diperhatikan, perintah if ini hampir sama dengan perintah if pada bahasa-bahasa tingkat tinggi, seperti Pascal, C, dan juga hampir sama dengan perintah if pada batch file-nya DOS. Pada bentuk pertama maupun bentuk kedua dari sintak diatas adalah statement

dalam blok if...fi akan dieksekusi apabila kondisi if terpenuhi. Dari kedua bentuk diatas dapat pula ditambahkan perintah untuk pengecekan kondisi dengan elif (else if).

8. INSTRUKSI TEST

Instruksi test digunakan untuk memeriksa kondisi dari sebuah ekspresi. Ekspresi terdiri dari factori dan operator yang dipisahkan oleh spasi. Hasil test akan memberikan nilai berupa status exit, yaitu 0 bila ekspresi sesuai, bila tidak maka hasilnya adalah tidak sama dengan nol.

- Operator untuk test

Operator	0 atau TRUE, jika
string1 = string2	Identical
string1!= string2	Not identical
-n string	String is not null
-z string	String is null

- Test untuk files dan direktori

Operator	0 atau TRUE, jika
-f namafile	File ada, file biasa
-d namafile	File ada, file adalah direktori
-r namafile	File dapat dibaca
-w namafile	File dapat ditulis
-x namafile	File adalah executable
-s namafile	File ada dan tidak kosong
-w namafile	File dapat ditulis

Untuk memudahkan pembacaan (readability), test dapat ditulis dengan :

[ekspresi]

Tanda [sebenarnya adalah nama lain dari test,bedanya [akan mencari kurung penutup] pada akhir ekspresi yang harus dipisahkan oleh spasi.

9. LOGICAL && DAN || (SHELL LEVEL)

Notasi && dan || digunakan untuk menggabungkan instruksi shell sebagai alternative dari if then else. Notasi && dan || sering ditemukan pada shell script system administrator untuk menjalankan routine dari system operasi.

- **instruksi1 && instruksi2**

shell akan mengeksekusi instruksi1, dan bila exit status instruksi1 adalah false, maka hasil dari AND tersebut sudah pasti sama dengan FALSE, sehingga instruksi2 tidak mempunyai pengaruh lagi. Oleh karna itu instruksi2 tidak dijalankan. Sebaliknya bila hasil instruksi1 adalah true/0, maka instruksi2 dijalankan.

- **instruksi1 || instruksi2**

shell akan mengeksekusi instruksi1, dan bila exit status adalah TRUE(0), hasil dari operasi OR tersebut sudah pasti menghasilkan TRUE, terlepas dari hasil eksekusi instruksi2. Oleh karena itu instruksi2 tidak perlu dijalankan.

Bila hasil instruksi1 adalah FALSE, maka instruksi2 akan dijalankan.

10. OPERATOR BILANGAN BULAT UNTUK TEST

Untuk membandingkan 2 buah bilangan, test memerlukan operator yang berbeda dengan string.

Operator	0 atau TRUE, jika
i1 -eq i2	Bilangan sama
i1 -ge i2	Lebih besar atau sama dengan
i1 -gt i2	Lebih besar
i1 -le i2	Lebih kecil atau sama dengan
i1 -lt i2	Lebih kecil
i1 -ne i2	Bilangan tidak sama

11. OPERATOR LOGICAL (TEST LEVEL)

Logical operator terdiri dari AND(-a), OR(-o), dan NOT(!). Operator ini menggabungkan hasil ekspresi sebagai berikut :

A	B	!A	A -a B	A -o B
False	False	True	False	False
False	True	True	False	True
True	False	False	False	True
true	true	false	true	true

12. KONSTRUKSI IF THEN ELSE IF

Bila status exit tidak sama dengan 0, maka kondisi menjadi FALSE dan instruksi setelah else akan dijalankan.

13. HITUNGAN ARITMATIKA

Untuk melakukan perhitungan aritmatik, shell tidak memiliki kemampuan built-in, tetapi meminta bantuan program lain yaitu `expr`. Program expr berfungsi untuk mengevaluasi suatu ekspresi baik itu perbandingan string atau operasi aritmatik sederhana.

Operator aritmatik yang disediakan expr antara lain:

Operator	KETERANGAN
+	Operator Penjumlahan contoh: <code>expr 1 + 1</code>
-	Operator Pengurangan contoh: <code>expr 10 - 9</code>
*	Operator Perkalian contoh: <code>expr 10 * 10</code>
/	Operator Pembagian contoh: <code>expr 10 / 2</code>
%	Operator Modulus contoh: <code>expr 15 % 3</code>

Selain digunakan untuk perhitungan aritmatik, perintah `expr` juga cukup handal untuk melakukan manipulasi string, untuk lebih jelas silahkan lihat halaman manual dari expr dengan mengetikkan `man expr`.

14. INSTRUKSI EXIT

Program dapat dihentikan (terminated/selesai) dengan intruksi exit. Sebagai nilai default program tersebut akan memberikan status exit 0.

15. KONSTRUKSI CASE

Pernyataan case memiliki fungsi yang sama dengan pernyataan if yaitu memeriksa suatu ekspresi dan melakukan suatu proses bila ekspresi tersebut bernilai benar, hanya saja pernyataan case memiliki kelebihan yaitu dapat menggunakan wildcards dalam ekspresi yang akan diperiksa.

Perintah case hampir sama dengan if-then-elif-else-fi karena dapat memilih dari banyak kondisi. Sebagian orang lebih suka menggunakan case...esac dibanding multi level-if karena lebih mudah dibaca. Case digunakan untuk menyederhanakan pemakaian if yang berantai, sehingga dengan case kondisi dapat dikelompokkan lebih jelas dan mudah untuk ditulis.

case variable in

```
match1)
instruksi1.1
instruksi1.2
..... ;;
match2 )
instruksi2.1
instruksi2.2
.....
;; *)
    instruksi3.1
instruksi3.2
..... ;;
esac
```


Case diakhiri dengan esac dan pada setiap kelompok instruksi diakhiri dengan ;;. Pada akhir pilihan yaitu *) yang berarti adalah “default”, bila kondisi tidak memenuhi pola sebelumnya.

16. KONSTRUKSI FOR

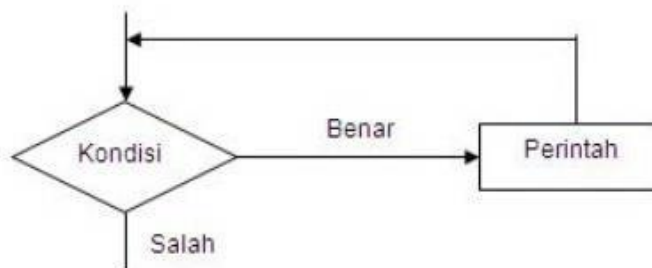
For digunakan untuk perulangan dengan menggunakan variable yang ada pada setiap perulangan akan diganti dengan nilai yang berada pada daftar (list).

```
for var in str1 str2 ....
strn
do      instruksi1      instruksi2
      .....
done
```

17. KONSTRUKSI WHILE

Pernyataan perulangan ini umumnya digunakan untuk melakukan perulangan dengan kondisi tertentu, di mana perulangan akan terus dilakukan dalam blok while selama ekspresi yang menjadi syarat masih dipenuhi (bernilai benar). Untuk ilustrasi, dapat dilihat pada gambar Sedangkan mengenai bentuk dari pernyataan ini adalah:

```
while <ekspresi>
do
    <perintah>
done
```



While digunakan untuk perulangan instruksi, yang umumnya dibatasi dengan suatu kondisi. Selama kondisi tersebut true, maka pengulangan akan terus dilakukan. Loop akan berhenti, bila kondisi false, atau program keluar dari blok while melalui exit atau break.

```
while kondisi do   instruksi1
instruksi2   .....
done
```

18. INSTRUKSI DUMMY

Instruksi dummy adalah instruksi yang tidak melakukan apa-apa, namun instruksi ini memberikan status exit 0 (TRUE). Oleh karena itu, instruksi dummy dapat digunakan sebagai kondisi forever pada loop (misalnya while).

Symbol instruksi dummy adalah → :

19. FUNGSI

Hampir sama dengan bahasa pemrograman tingkat tinggi, pada shell script juga dikenal istilah fungsi. Dimana dengan adanya fungsi kita dapat membagi kode kita ke dalam sub-sub yang lebih kecil. Hal ini sangat berguna jika kita membangun sebuah program shell script yang cukup kompleks.

Fungsi adalah program yang dapat dipanggil oleh program lainnya dengan menggunakan notasi NamaFungsi(). Fungsi memberikan exit status (\$?) yang dinyatakan dengan **return nr**, atau nilai 0 sebagai default. Membuat fungsi diawali dengan nama fungsi, parameter, kemudian blok program yang dinyatakan dalam {.....}. Contoh :

```
F1 ( )
{
    .....
    return 1
}
```

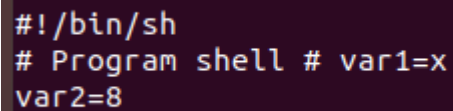
Variabel dapat di definisikan dalam fungsi sebagai variabel local atau global. Hal yang perlu diperhatikan, nama variable yang digunakan dalam sebuah fungsi, jangan sampai bentrok dengan nama variable yang sama di luar fungsi, sehingga tidak terjadi isi variable berubah. Login sebagai user, bukalah Console Terminal dan lakukan percobaan-percobaan dibawah ini kemudian selesaikan soal-soal latihan !

Percobaan 1 : Membuat Shell Script

```
$ vi prog01.sh
```

```
#!/bin/sh
```

```
# Program shell # var1=x  
var2=8
```



```
#!/bin/sh  
# Program shell # var1=x  
var2=8
```

Analisis

Gambar di atas merupakan perintah untuk membuat shell script dengan membuat file prog01.sh menggunakan editor vi. Perintah **\$ vi prog01.sh** berfungsi untuk mengedit file konfigurasi sistem dan membuat script. Kemudian tekan i untuk insert script. Script berikut

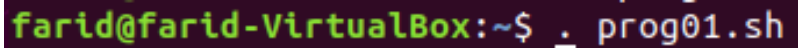
```
#!/bin/sh
```

```
# Program shell
```

var1=x var2=8 berfungsi untuk mendefinisikan var1 sama dengan x dan var2 sama dengan 8. tekan esc pada keyboard lalu setelah tulisan INSERTnya hilang ketikan :wq berfungsi untuk menyimpan dan keluar dari file tersebut.

1. Untuk menjalankan shell, gunakan notasi TITIK di depan nama program

```
$ . prog01.sh
```



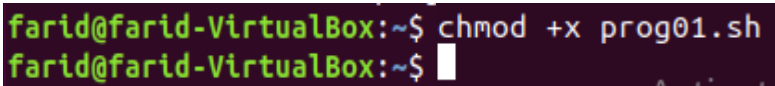
```
farid@farid-VirtualBox:~$ . prog01.sh
```

Analisis

Gambar di atas merupakan perintah untuk menjalankan shell, gunakan notasi TITIK di depan nama program. Ketikkan perintah di atas, tapi prog01.sh ini tidak menampilkan teks/string seperti menggunakan perintah echo, maka pada saat prog01.sh dijalankan tidak akan ditampilkan apa-apa.

2. Untuk menjalankan shell, dapat juga membuat executable file dan dieksekusi dari current directory

```
$ chmod +x prog01.sh
```



```
farid@farid-VirtualBox:~$ chmod +x prog01.sh
farid@farid-VirtualBox:~$
```

Analisis

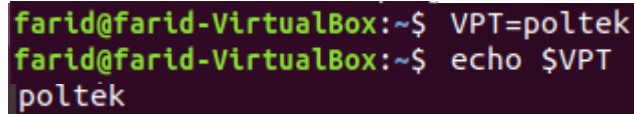
Gambar di atas merupakan perintah untuk menjalankan shell, dapat juga membuat executable file dan dieksekusi dari current directory. Setelah mengetikkan perintah di atas, prog01.sh tidak akan menampilkan apa-apa karena isi dari script prog01.sh hanya mendefinisikan var1 dan var2 tanpa ada proses menampilkan.

Percobaan 2 : Variable

1. Contoh menggunakan variable pada shell interaktif

```
$ VPT=poltek
```

```
$ echo $VPT =
```



```
farid@farid-VirtualBox:~$ VPT=poltek
farid@farid-VirtualBox:~$ echo $VPT
poltek
```

Analisis

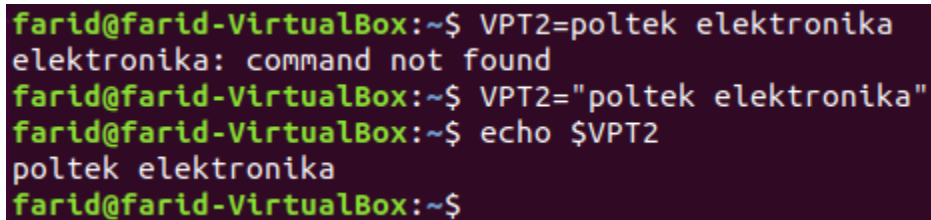
Gambar di atas merupakan contoh menggunakan variable pada shell interaktif. Pada shell aktif untuk menampilkan variabel pertama definisikan nama variabel tersebut, misalnya variabel VPT berisi poltek. Dan untuk menampilkan atau memanggil variabel tersebut dapat menggunakan echo \$VPT maka setelah di enter isi dari variabel VPT akan tampil yaitu output dengan tulisan poltek.

2. Pemisahan 2 kata dengan spasi menandakan eksekusi 2 buah intruksi. Karakter \$ harus ada pada awal nama variable untuk melihat isi variable tersebut, jika tidak, maka echo akan mengambil parameter tersebut sebagai string.

```
$ VPT2=poltek elektronika (terdapat pesan error)
```

```
$ VPT2="poltek elektronika"
```

```
$ echo $VPT2
```



```
farid@farid-VirtualBox:~$ VPT2=poltek elektronika
elektronika: command not found
farid@farid-VirtualBox:~$ VPT2="poltek elektronika"
farid@farid-VirtualBox:~$ echo $VPT2
poltek elektronika
farid@farid-VirtualBox:~$
```

Analisis

Gambar di atas merupakan pemisah 2 kata dengan spasi menandakan eksekusi 2 buah intruksi. Terdapat beberapa perbedaan untuk program di atas. Untuk isi dari sebuah variabel apabila terdiri dari dua buah kata maka tidak bisa secara langsung dipisahkan dengan spasi jadi terdapat pesan error. Untuk membuat variabel dengan 2 kata harus diapit oleh tanda kutip dua (“”). Kemudian pada perintah selanjutnya untuk menampilkan variabel VPT2 ini harus diawali dengan tanda \$ jika tidak, maka echo akan mengambil parameter tersebut sebagai string.

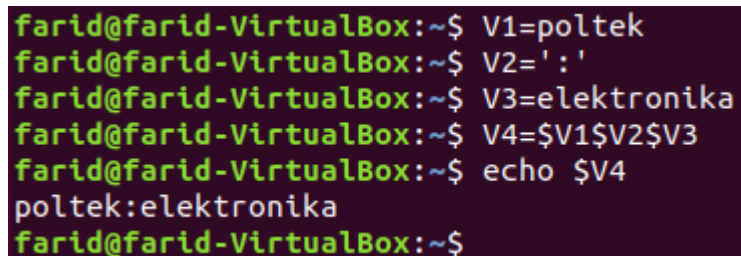
3. Menggabungkan dua variable atau lebih

```
$ V1=poltek $ V2=':'
```

```
$ V3=elektronika
```

```
$ V4=$V1$V2$V3
```

```
$ echo $V4
```



```
farid@farid-VirtualBox:~$ V1=poltek
farid@farid-VirtualBox:~$ V2=':'
farid@farid-VirtualBox:~$ V3=elektronika
farid@farid-VirtualBox:~$ V4=$V1$V2$V3
farid@farid-VirtualBox:~$ echo $V4
poltek:elektronika
farid@farid-VirtualBox:~$
```

Analisis

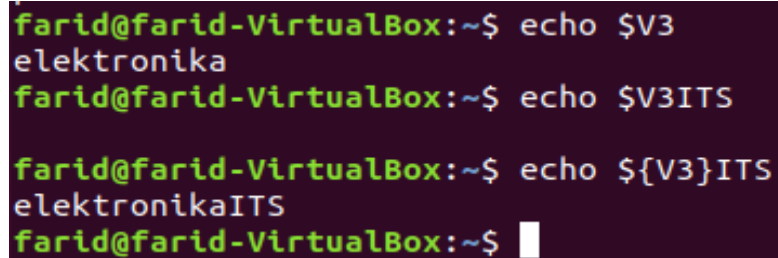
Gambar di atas merupakan penggabungan dua variable atau lebih. Untuk menggabungkan dua variabel atau lebih selain dapat ditulis secara langsung bisa juga didefinisikan lagi kedalam variabel lain contohnya pada gambar diatas tiga buah variabel didefinisikan di variabel V4 maka untuk menampilkan variabel V1 V2 V3 cukup memanggil variabel V4 saja.

4. Menggabungkan isi variable dengan string yang lain. Jika digabungkan dengan nama variable yang belum didefinisikan (kosong) maka instruksi echo menghasilkan string kosong. Untuk menghindari kekeliruan, nama variable perlu diproteksi dengan { } dan kemudian isi variable tersebut digabung dengan string.

```
$ echo $V3
```

```
$ echo $V3ITS
```

```
$ echo ${V3}ITS
```

A terminal window with a dark purple background and green text. It shows three commands and their outputs. The first command is 'echo \$V3' which outputs 'elektronika'. The second command is 'echo \$V3ITS' which outputs an empty string. The third command is 'echo \${V3}ITS' which outputs 'elektronikaITS'.

```
farid@farid-VirtualBox:~$ echo $V3
elektronika
farid@farid-VirtualBox:~$ echo $V3ITS

farid@farid-VirtualBox:~$ echo ${V3}ITS
elektronikaITS
farid@farid-VirtualBox:~$
```

Analisis

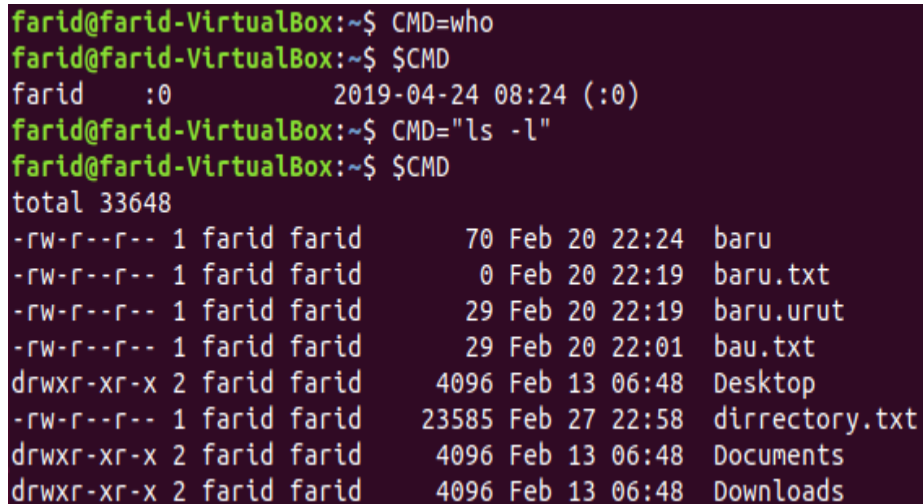
Gambar di atas merupakan penggabungan isi variable dengan string yang lain. Untuk menggabungkan sebuah variabel dengan string lain tidak dapat secara langsung seperti pada perintah 2 diatas. Jika penulisan nya seperti \$V3ITS maka instruksi echo akan menampilkan sting kosong karena variabel tersebut dianggap sebagai variabel yang belum terdefinisikan. Untuk menggabungkan isi variabel dengan string lain digunakan tanda { } untuk memisahkan variabel dengan string lain tersebut.

5. Variable dapat berisi instruksi, yang kemudian bila dijadikan instruksi tersebut akan dieksekusi

```
$ CMD=who $ $CMD
```

```
$ CMD="ls -l"
```

```
$ $CMD
```



```
farid@farid-VirtualBox:~$ CMD=who
farid@farid-VirtualBox:~$ $CMD
farid      :0                2019-04-24 08:24 (:0)
farid@farid-VirtualBox:~$ CMD="ls -l"
farid@farid-VirtualBox:~$ $CMD
total 33648
-rw-r--r-- 1 farid farid      70 Feb 20 22:24 baru
-rw-r--r-- 1 farid farid       0 Feb 20 22:19 baru.txt
-rw-r--r-- 1 farid farid     29 Feb 20 22:19 baru.urut
-rw-r--r-- 1 farid farid     29 Feb 20 22:01 bau.txt
drwxr-xr-x 2 farid farid   4096 Feb 13 06:48 Desktop
-rw-r--r-- 1 farid farid  23585 Feb 27 22:58 dirrectory.txt
drwxr-xr-x 2 farid farid   4096 Feb 13 06:48 Documents
drwxr-xr-x 2 farid farid   4096 Feb 13 06:48 Downloads
```

Analisis

Gambar di atas merupakan variabel berisi instruksi, yang kemudian bila dijadikan instruksi tersebut akan dieksekusi. Variabel dapat berisi instruksi. Seperti pada contoh diatas variabel CMD diisi oleh instruksi who, maka apabila variabel CMD dipanggil instruksi who yang terdapat dalam variabel CMD akan dieksekusi. Begitu juga dengan variabel CMD yang diisi dengan perintah ls -l, dan tampil dengan output menampilkan file file pada direktori.

6. Membuat shell script, contoh **prog01.sh** sebagai berikut :

```
$ vi prog01.sh
```

```
#!/bin/sh
```

```
V1=poltek V2=':' V3=elektronika
```

```
echo "Pemrograman shell" echo
```

```
$V1$V2$V3 V3=ITS echo $V1$2 di
```

```
$V3
```



```
#!/bin/sh
V1=poltek V2=':' V3=elektronika
echo "Pemrograman shell" echo
echo $V1$V2$V3
V3=ITS
echo $V1$V2$V3
```

Analisis

Gambar di atas merupakan pengeditan shell script pada percobaan pertama. Shell script dari modifikasi prog01.sh ini berisi 3 buah variabel yaitu V1 V2 V3. Pada awalnya variabel V3 berisi dengan string elektronika namun kemudian variabel V3 diubah dengan isi ITS. Maka apabila ditampilkan echo pertama akan menampilkan pemrograman shell, kemudian echo kedua akan menampilkan poltek:elektronika sedangkan echo ketiga akan menampilkan poltek: di ITS karena V3 sebelumnya telah mengalami perubahan isi.

7. Menjalankan prog01.sh

```
$ . prog01.sh
$ prog01.sh (TERDAPAT PESAN ERROR)
$ ./prog01.sh (TERDAPAT PESAN ERROR)
$ chmod +x prog01.sh
$ ./prog01.sh
```

```
farid@farid-VirtualBox:~$ vi prog01.sh
farid@farid-VirtualBox:~$ . prog01.sh
farid@farid-VirtualBox:~$ prog01.sh
prog01.sh: command not found
farid@farid-VirtualBox:~$ ./prog01.sh
```

Analisis

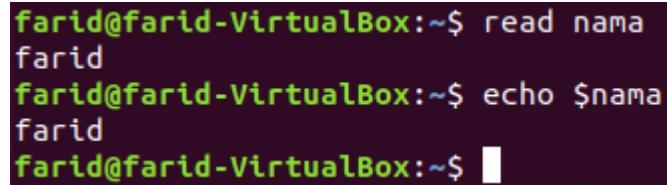
Gambar di atas merupakan cara untuk menjalankan prog01.sh. Untuk memanggil prog01.sh ini dapat dilakukan dengan beberapa cara seperti diatas. Pemanggilan kedua dan ketiga gagal, karena pemanggilan tersebut hanya untuk file executable.

Percobaan 3 : Membaca Keyboard

1. Menggunakan intruksi read

```
$ read nama Pebri
```

```
$ echo $nama
```



```
farid@farid-VirtualBox:~$ read nama
farid
farid@farid-VirtualBox:~$ echo $nama
farid
farid@farid-VirtualBox:~$
```

Analisis

Gambar di atas merupakan cara untuk membaca keyboard. Untuk membaca input dari keyboard dapat menggunakan instruksi read. Program diatas variabel nama akan dinput melalui keyboard yaitu Pebri, pada saat dipanggil maka variabel nama tersebut kan menampilkan Pebri.

2. Membaca nama dan alamat dari keyboard.

```
$ vi prog02.sh
```

```
#!/bin/sh
```

```
# prog02.sh
```

```
# membaca nama dan alamat echo
```

```
"Nama Anda :" read nama echo
```

```
"Alamat : " read alamat
```

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

```
#!/bin/sh
#prog02.sh
#membaca nama dan alamat
echo "Nama Anda : "
read nama
echo "Alamat : "
read alamat
echo "Kota : "
read kota

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

Analisis

Gambar di atas merupakan pembuatan file prog02.sh. Sama halnya dengan percobaan pada point sebelumnya, namun percobaan membaca variabel dari keyboard ini dilakukan melalui shell script prog02.sh. 3 buah variabel nama alamat dan kota merupakan input dari keyboard menggunakan perintah read dan akan di eksekusi pada percobaan berikutnya.

3. Eksekusi program prog02.sh

```
$ . prog02.sh Nama
```

```
Anda : Fana
```

```
Alamat : Perum.green
```

```
view
```

```
kota: malang
```

```
Hasil adalah : Fana,perum.green View,malang
```

```
farid@farid-VirtualBox:~$ . prog02.sh
Nama Anda :
farid
Alamat :
jl. dewa
kota :
jakarta

Hasil adalah : farid, jl. dewa di jakarta
```

Analisis

Gambar di atas merupakan perintah untuk menjalankan proses membaca keyboard. Pada saat prog02.sh dieksekusi akan tampil tulisan nama anda : kemudian diikuti dengan perintah memasukan input dari variabel nama dengan posisi satu enter. Begitu juga untuk variabel alamat dan kota. Kemudian setelah menginputkan variabel kota maka akan tampil Hasil adalah : dengan variabel nama, variabel alamat dan diikuti dengan variabel kota.

5. Variable kosong adalah variabel yang tidak mempunyai nilai.

```
$ read nama <CR>
```

```
$ echo $nama
```

```
$ A=
```

```
$ B=""
```

```
$ C=$A$B
```

```
$ echo $c
```

```
farid@farid-VirtualBox:~$ read nama
<CR>
farid@farid-VirtualBox:~$ echo $nama
<CR>
farid@farid-VirtualBox:~$ A=
farid@farid-VirtualBox:~$ B=""
farid@farid-VirtualBox:~$ C=$A$B
farid@farid-VirtualBox:~$ echo $C

farid@farid-VirtualBox:~$
```

Analisis

Gambar di atas merupakan perintah untuk membaca keyboard. Seluruh perintah dari variabel diatas merupakan variabel kosong atau tidak mempunyai nilai. Maka pada saat variabel dipanggil tidak akan menampilkan apapun.

6. Variable dapat didistribusikan dengan hasil eksekusi dari sebuah instruksi.

```
$ pwd
```

```
$DIR='pwd'
```

```
$echo $DIR
```

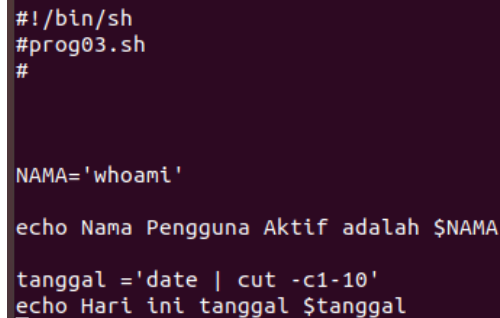
```
farid@farid-VirtualBox:~$ pwd
/home/farid
farid@farid-VirtualBox:~$ DIR='pwd'
farid@farid-VirtualBox:~$ echo $DIR
pwd
farid@farid-VirtualBox:~$
```

Analisis

Gambar di atas merupakan perintah untuk membaca keyboard. Untuk mengisi variabel dengan substitusi hasil eksekusi dari sebuah eksekusi instruksi harus menggunakan sepasang back quate (`). Misalnya pada contoh diatas variabel DIR didistribusikan dengan hasil eksekusi dari perintah pwd. Maka saat variabel DIR dipanggil akan tampil hasil dari pwd tersebut.

7. Buatlah shell script **prog03.sh** kemudian eksekusi!

```
$ vi prog03.sh
#!/bin/sh
# prog03.sh #
NAMA='whoami'
echo Nama pengguna Aktif adalah $NAMA
tanggal='date | cut -c1-10' echo hari ini
anggal $tanggal
```



```
#!/bin/sh
#prog03.sh
#

NAMA='whoami'

echo Nama Pengguna Aktif adalah $NAMA

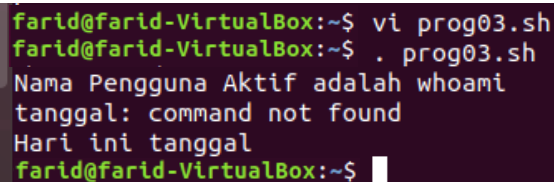
tanggal ='date | cut -c1-10'
echo Hari ini tanggal $tanggal
```

Analisis

Gambar di atas merupakan perintah untuk membuat file Prog03.sh untuk menampilkan variabel dari hasil eksekusi perintah whoami dan date. Variabel whoami didefinisikan dalam variable NAMA sedangkan date didefinisikan dengan variabel tanggal dan akan di eksekusi pada percobaan berikutnya

9. Eksekusi prog03.sh

```
$ . prog03.sh
```



```
farid@farid-VirtualBox:~$ vi prog03.sh
farid@farid-VirtualBox:~$ . prog03.sh
Nama Pengguna Aktif adalah whoami
tanggal: command not found
Hari ini tanggal
farid@farid-VirtualBox:~$
```

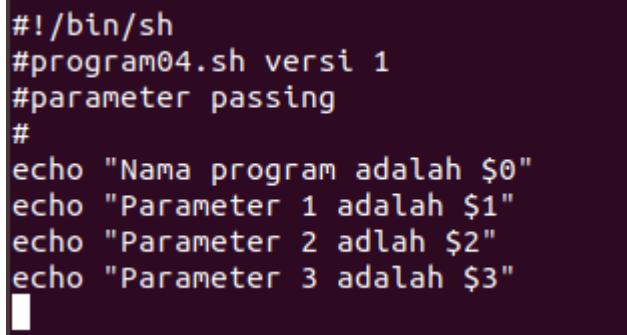
Analisis

Gambar di atas merupakan hasil dari eksekusi shell script prog03.sh. nama pengguna aktif berisi variabel NAMA yaitu hasil eksekusi dari perintah whoami. Kemudian Hari ini tanggal akan diikuti dengan variabel tanggal dengan isi hasil dari instruksi date.

Percobaan 4 : Parameter

1. Membuat shell script **prog04.sh**

```
$ vi prog04.sh
#!/bin/sh
# prog04.sh versi 1 #
Parameter passing
#
echo "Nama program adalah $0" echo
"Parameter 1 adalah $1" echo "Parameter 2
adalah $2" echo "Parameter 3 adalah $3"
```



```
#!/bin/sh
#program04.sh versi 1
#parameter passing
#
echo "Nama program adalah $0"
echo "Parameter 1 adalah $1"
echo "Parameter 2 adlah $2"
echo "Parameter 3 adalah $3"
█
```

Analisis

Gambar di atas merupakan perintah untuk membaca parameter. Perintah `$ vi prog04.sh` berfungsi untuk membuat shell script `prog04.sh`. script berikut :

```
# prog04.sh versi 1 #  
Parameter passing  
#  
echo "Nama program adalah $0"  
echo "Parameter 1 adalah $1" echo  
"Parameter 2 adalah $2" echo  
"Parameter 3 adalah $3"
```

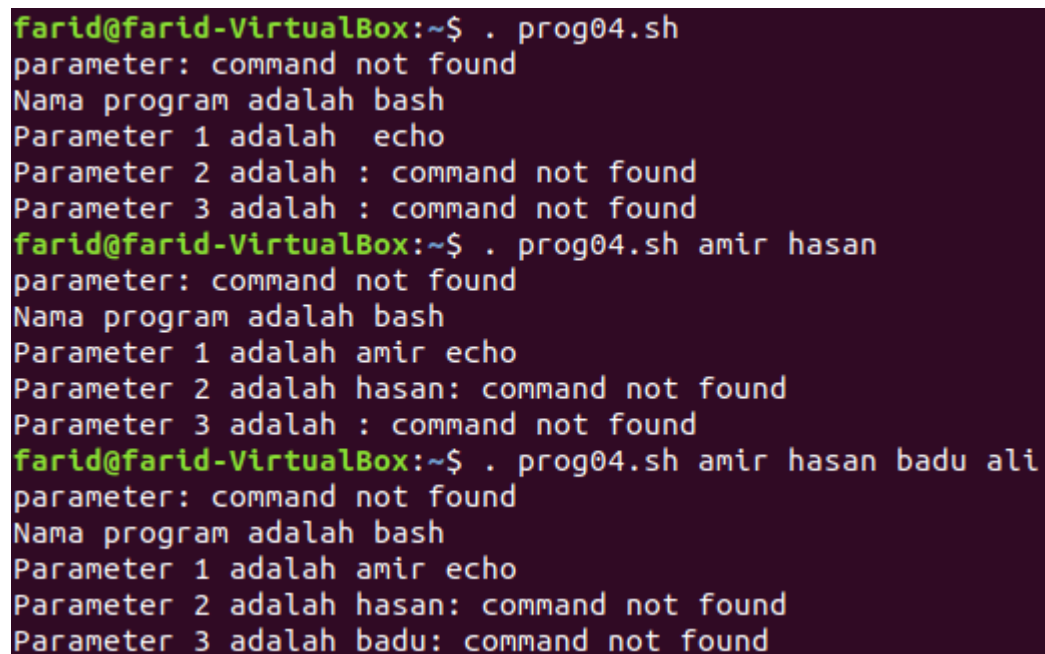
merupakan contoh dari pembuatan parameter dengan 4 buah parameter \$0 hingga \$3 dan akan di eksekusi pada percobaan berikutnya.

2. Eksekusi **prog04.sh** tanpa parameter, dengan 2 parameter dan dengan 4 parameter!

```
$ . prog04.sh
```

```
$ . prog04.sh amir hasan
```

```
$ . prog04.sh amir hasan badu ali
```



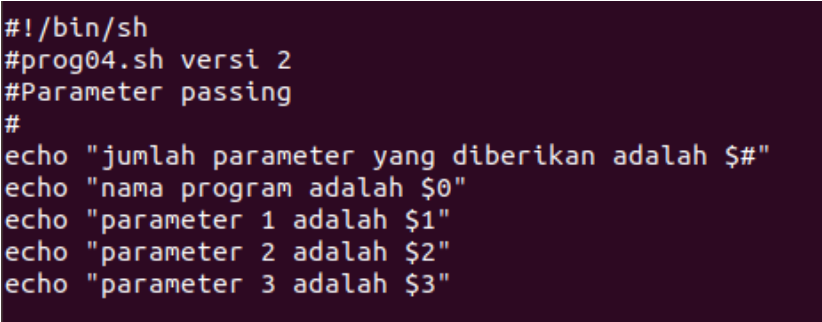
```
farid@farid-VirtualBox:~$ . prog04.sh  
parameter: command not found  
Nama program adalah bash  
Parameter 1 adalah echo  
Parameter 2 adalah : command not found  
Parameter 3 adalah : command not found  
farid@farid-VirtualBox:~$ . prog04.sh amir hasan  
parameter: command not found  
Nama program adalah bash  
Parameter 1 adalah amir echo  
Parameter 2 adalah hasan: command not found  
Parameter 3 adalah : command not found  
farid@farid-VirtualBox:~$ . prog04.sh amir hasan badu ali  
parameter: command not found  
Nama program adalah bash  
Parameter 1 adalah amir echo  
Parameter 2 adalah hasan: command not found  
Parameter 3 adalah badu: command not found
```


Analisis

Gambar di atas merupakan perintah membaca parameter. Apabila prog04.sh hanya di eksekusi tanpa diikuti dengan parameter maka akan menampilkan nama program dengan \$0 adalah bash sedangkan parameter 1 2 dan 3 tidak akan berisi parameter. Eksekusi prog04.sh kedua dan ketiga diikuti dengan beberapa parameter. Pertama hanya ada dua parameter yaitu pebri dan ramdani maka parameter yang akan diisi hanya parameter 1 dan 2. Sedangkan selanjutnya diikuti dengan 4 buah parameter. Sehingga nama tanid tidak akan dimasukkan kedalam parameter secara merurutan.

3. Membuat shell script **prog04.sh** versi 2 dengan memberikan jumlah parameter

```
$ vi prog04.sh
#!/bin/sh
# prog04.sh versi 2 #
Parameter passing
#
echo "Jumlah parameter yang diberikan adalah $# " echo
"Nama program adalah $0" echo "Parameter 1 adalah $1"
echo "Parameter 2 adalah $2" echo
"Parameter 3 adalah $3"
```



```
#!/bin/sh
#prog04.sh versi 2
#Parameter passing
#
echo "jumlah parameter yang diberikan adalah $# "
echo "nama program adalah $0"
echo "parameter 1 adalah $1"
echo "parameter 2 adalah $2"
echo "parameter 3 adalah $3"
```

Analisis

Gambar di atas merupakan perintah memodifikasi shell script pada program parameter prog04.sh. pada text editor vi diisikan script berikut

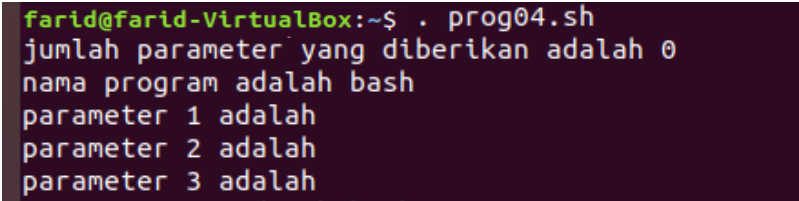
```
#!/bin/sh
# prog04.sh versi 2 #
Parameter passing
#
echo "Jumlah parameter yang diberikan adalah
$#" echo "Nama program adalah $0" echo
"Parameter 1 adalah $1" echo "Parameter 2
adalah $2" echo "Parameter 3 adalah $3"
```

sehingga parameter yang diberikan didefinisikan dengan simbol \$#, dan akan di eksekusi pada percobaan berikutnya.

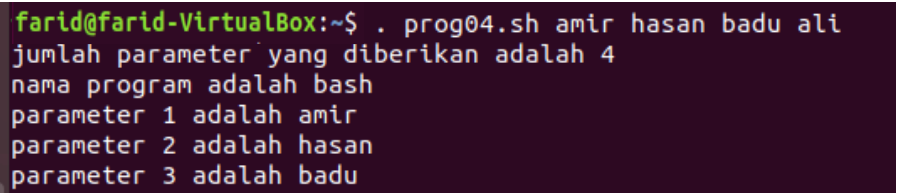
4. Eksekusi **prog04.sh** tanpa parameter dan dengan 4 parameter

```
$ . prog04.sh
```

```
$ . prog04.sh pebri tanib tanic tanid
```



```
farid@farid-VirtualBox:~$ . prog04.sh
jumlah parameter yang diberikan adalah 0
nama program adalah bash
parameter 1 adalah
parameter 2 adalah
parameter 3 adalah
```



```
farid@farid-VirtualBox:~$ . prog04.sh amir hasan badu ali
jumlah parameter yang diberikan adalah 4
nama program adalah bash
parameter 1 adalah amir
parameter 2 adalah hasan
parameter 3 adalah badu
```

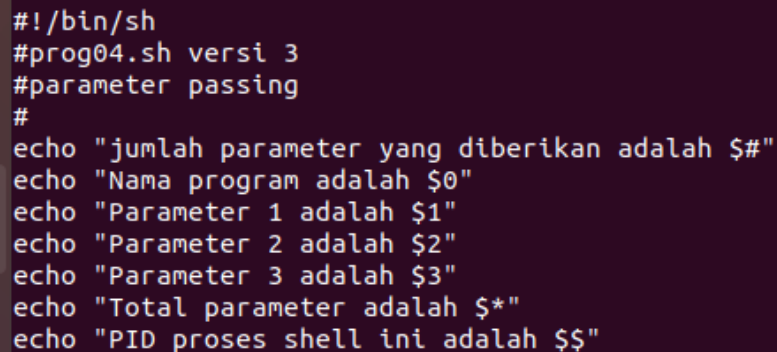
Analisis

Gambar di atas merupakan perintah membaca parameter. Eksekusi pertama prog04.sh tidak diisi dengan parameter maka \$# akan menampilkan 0, sedangkan

eksekusi prog04.sh diikuti dengan pebri ramdani coba lagi dihitung sebagai 4 parameter, dan tampil dengan jumlah parameternya ditampilkan.

5. Membuat shell script **prog04.sh** versi 3 dengan menambahkan total parameter dan nomor proses id

```
$ vi prog04.sh
#!/bin/sh
# prog04. Sh versi 3
# Parameter passing
#
echo "Jumlah parameter yang diberikan adalah $# " echo
"Nama program adalah $0" echo "Parameter 1 adalah $1"
echo "Parameter 2 adalah $2" echo "Parameter 3 adalah $3"
echo "Total parameter adalah $*"
echo "PID proses shell ini adalah $$"
```



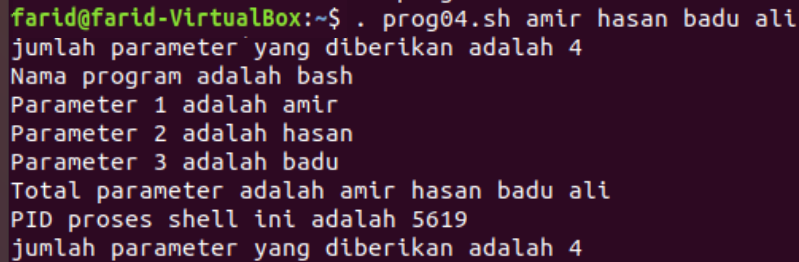
```
#!/bin/sh
#prog04.sh versi 3
#parameter passing
#
echo "jumlah parameter yang diberikan adalah $# "
echo "Nama program adalah $0"
echo "Parameter 1 adalah $1"
echo "Parameter 2 adalah $2"
echo "Parameter 3 adalah $3"
echo "Total parameter adalah $*"
echo "PID proses shell ini adalah $$"
```

Analisis

Gambar di atas merupakan perintah membaca parameter. Pada prog04.sh versi 3 ini diberikan dengan nomor proses id yang didefinisikan dengan \$. Sedangkan untuk menampilkan seluruh parameter yang diinputkan menggunakan instruksi \$*, dan akan di eksekusi pada percobaanberikutnya.

6. Eksekusi **prog04.sh** dengan 4 parameter

```
$ . prog04.sh pebri ramdani coba lagi
```



```
farid@farid-VirtualBox:~$ . prog04.sh amir hasan badu ali
jumlah parameter yang diberikan adalah 4
Nama program adalah bash
Parameter 1 adalah amir
Parameter 2 adalah hasan
Parameter 3 adalah badu
Total parameter adalah amir hasan badu ali
PID proses shell ini adalah 5619
jumlah parameter yang diberikan adalah 4
```

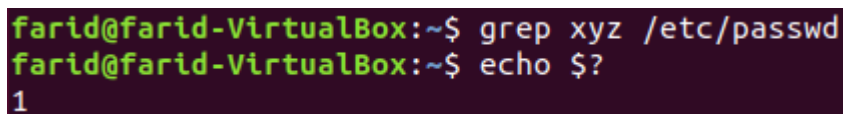
Analisis

Gambar di atas merupakan perintah membaca parameter. Total parameter adalah merupakan hasil untuk menampilkan pebri ramdani coba lagi dari instruksi \$*. Sedangkan PID proses shell ini adalah 9072 dengan instruksi \$\$, dan menampilkan program sebelumnya dengan menampilkan juga no PID yang digunakan.

Percobaan 5 : Status Exit

1. String tidak ditemukan, maka status exit adalah 1

```
$ grep xyz / etc/ passwd $ echo $?
```



```
farid@farid-VirtualBox:~$ grep xyz /etc/passwd
farid@farid-VirtualBox:~$ echo $?
1
```

Analisis

Gambar di atas merupakan perintah status exit. Apabila string yang tidak ditemukan pada instruksi grep seperti diatas, akan ditampilkan status exit dengan perintah echo \$? adalah 1, artinya tidak ditemukan.

2. String ditemukan, maka status exit adalah 0

```
$ grep <user> /etc/passwd $ echo $?
```

```
farid@farid-VirtualBox:~$ grep farid /etc/passwd
farid:x:1000:1000:Farid Aziz,1,1231123,123112:/home/farid:/bin/bash
farid@farid-VirtualBox:~$ echo $?
0
```

Analisis

Gambar di atas merupakan perintah status exit. Pada contoh diatas ini string pebri.ramdani ditemukan dalam directory /etc/passwd maka apabila echo \$? dipanggil akan menampilkan status exit 0, dan artinya string ditemukan.

Percobaan 6 : Konstruksi if

1. grepInstruksi dengan exit status 0

```
$ who
```

```
$ who | grep <user> $ echo $?
```

```
farid@farid-VirtualBox:~$ who
farid      :0                2019-04-24 08:24 (:0)
farid@farid-VirtualBox:~$ who | grep farid
farid      :0                2019-04-24 08:24 (:0)
farid@farid-VirtualBox:~$ echo $
$
```

Analisis

Gambar di atas merupakan perintah konstruksi if. Masih merupakan bahasan status exit dari string pebri.ramdani dari instruksi who, maka status exitnya adalah 0 dan tampil dengan menampilkan hasil yang berkaitan dengan perintah who.

2. If membandingkan exit status dengan 0, bila sama, maka blok program masuk ke dalam blok then-fi

```
$ if [ $? = 0 ]
```

```
> then
```

```
> echo "Pemakai tersebut sedang aktif"
```

```
> fi
```

```
farid@farid-VirtualBox:~$ if [ $? = 0 ]; then echo "Pemakai terebut sedang aktif";  
fi  
Pemakai terebut sedang aktif  
farid@farid-VirtualBox:~$
```

Analisis

Gambar di atas merupakan perintah konstruksi if. Perintah if ini adalah untuk membandingkan status exit dengan 0, bila sama, maka program masuk ke dalam blok then if yaitu Pemakai tersebut sedang aktif, karena perintah sebelumnya user pebri ramdani sedang aktif hasilnya 0, apabila dengan if \$? = 1 artinya user tidak aktif.

3. Nomor (1) dan (2) diatas dapat disederhanakan dengan

```
$ if who | grep <user> >/dev/null  
> then  
> echo okay  
> fi
```

```
farid@farid-VirtualBox:~$ if who|grep farid /dev/null  
> then  
> echo okay  
> fi  
farid@farid-VirtualBox:~$
```

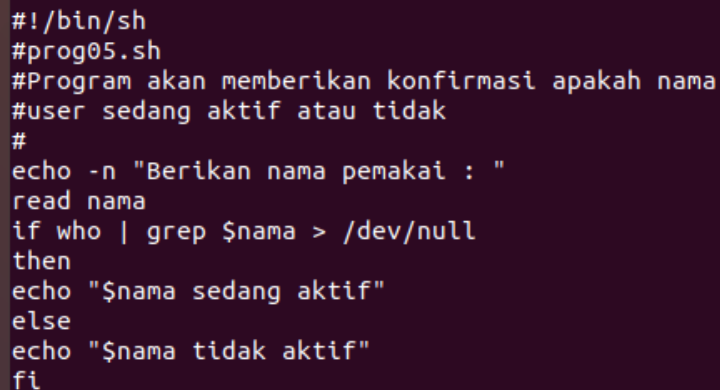
Analisis

Gambar di atas merupakan perintah konstruksi if. Pada percobaan ini merupakan penyederhanaan dari nomor 1 dan 2. Dengan instruksi if who dengan grep pebri.ramdani akan dibelokan ke /dev/null maka apabila perintahnya berstatus exit 0 akan ditampilkan okay.

Percobaan 7 : Konstruksi if then else

1. Membuat shell script **prog05.sh**

```
$ vi prog05.sh
#!/bin/sh
# prog05.sh
# Program akan memberikan konfirmasi apakah nama
# user sedang aktif atau tidak
#
echo -n "Berikan nama pemakai : " read
nama if who | grep $nama > /dev/null then
echo "$nama sedang aktif" else echo "$nama
tidak aktif" fi
```



```
#!/bin/sh
#prog05.sh
#Program akan memberikan konfirmasi apakah nama
#user sedang aktif atau tidak
#
echo -n "Berikan nama pemakai : "
read nama
if who | grep $nama > /dev/null
then
echo "$nama sedang aktif"
else
echo "$nama tidak aktif"
fi
```

Analisis

Gambar di atas merupakan perintah konstruksi if then else. Perintah `vi prog05.sh` digunakan untuk membuat shell script `prog05.sh`. isi dari program ini yaitu lanjutan dari percobaan sebelumnya tentang memberi konfirmasi apakah user sedang aktif atau tidak.

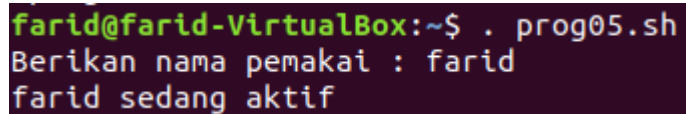
Konstruksi if then else ini digunakan pada script shell `prog05.sh` `grep` dari variabel `$nama` akan dibelokan ke `/dev/null`. Kemudian then apabila aktif atau else jika nama tidak aktif dan ditutup dengan `fi` dan akan di eksekusi pada percobaan berikutnya.

1. Jalankan **prog05.sh**, masukan nama pemakai aktif yang tampil pada instruksi who dan coba juga untuk nama pemakai yang tidak aktif

```
$ who
```

```
$ . prog05.sh [nama=<user>]
```

```
$ . prog05.sh [nma=studentOS]
```

A terminal window screenshot with a dark background. The prompt is 'farid@farid-VirtualBox:~\$'. The user has entered '. prog05.sh'. The script outputs 'Berikan nama pemakai : farid' and then 'farid sedang aktif' on the next line.

```
farid@farid-VirtualBox:~$ . prog05.sh
Berikan nama pemakai : farid
farid sedang aktif
```

Analisis

Gambar di atas merupakan perintah menampilkan konstruksi if then else. Perintah who ini menampilkan user aktif yaitu pebri.ramdani dan user yang tidak aktif yaitu student. Jika prog05.sh diisi dengan pebri.ramdani, maka akan ditampilkan pebri.ramdani sebagai variabel nama yaitu pebri.ramdani sedang aktif. Sedangkan jika prog05.sh diisi dengan nama pemakai student maka instruksi ini dinyatakan tidak aktif karena user yang aktif adalah pebri.ramdani.