

LAPORAN PRAKTIKUM SISTEM OPERASI

Praktikum 6b

Oleh:
FARID AZIZ WICAKSONO
NIM. 1841720094



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

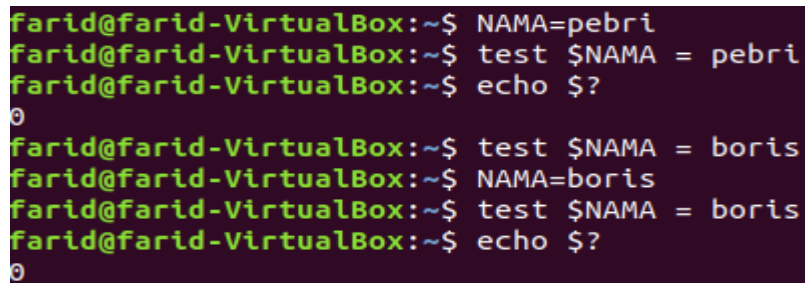
Percobaan 8 : Instruksi Test

1. Menggunakan instruksi test, perhatikan spasi antara

```
$ NAMA=pebri
```

```
$ test $NAMA = pebri $ echo $?
```

```
$ test $ NAMA = boris $ echo $?
```



```
farid@farid-VirtualBox:~$ NAMA=pebri
farid@farid-VirtualBox:~$ test $NAMA = pebri
farid@farid-VirtualBox:~$ echo $?
0
farid@farid-VirtualBox:~$ test $NAMA = boris
farid@farid-VirtualBox:~$ NAMA=boris
farid@farid-VirtualBox:~$ test $NAMA = boris
farid@farid-VirtualBox:~$ echo $?
0
```

Analisis

Gambar di atas merupakan perintah instruksi test. Variabel NAMA diisi dengan pebri. Kemudian untuk memanggil variabel NAMA dengan perintah test dengan \$NAMA didfinisikan pebri maka status exit bernilai 0. Sedangkan apabila berisi boris status exit berisi 1 artinya nama boris belum diinputkan, maka apabila dipanggil nama boris tidak ditemukan.

2. Aplikasi test dengan konstruksi if

```
$ vi prog06.sh
```

```
#!/bin/sh # prog06.sh echo -n
```

```
"NAMA = " read NAMA if test
```

```
"$NAMA" = pebri then
```

```
echo "Selamat Datang $NAMA" else
```

```
echo "Anda bukan pebri, sorry!" fi
```

```
#!/bin/sh
#prog06.sh
echo -n "NAMA = "
read NAMA
if test "$NAMA" = amir
then
echo "Selamat Datang $NAMA"
else
echo "Anda bukan amir, sorry!"
fi
```

Analisis

Gambar di atas merupakan perintah instruksi test. Perintah vi prog06.sh berfungsi

untuk membuat program dengan script shell berikut **#!/bin/sh**

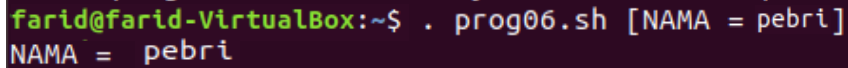
```
# prog06.sh echo
-n "NAMA = "
read NAMA
if test "$NAMA" = pebri then
echo "Selamat Datang $NAMA" else
echo "Anda bukan pebri, sorry!" fi
```

Sama halnya dengan point sebelumnya, yaitu aplikasi test, namun pada shell script prog06.sh. Variabel nama didefinisikan sebagai pebri. Dengan perintah if apabila variabel nama berisi pebri maka akan ditampilkan Selamat datang pebri sedangkan apabila bukan (else) maka akan ditampilkan Anda bukan pebri, sorry !! script fi untuk mengakhiri program.

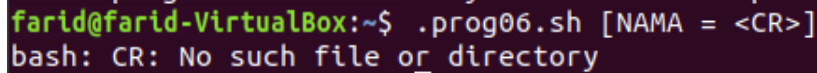
3. Jalankan program prog06.sh dengan memasukan NAMA = amir dan NAMA = <CR> perhatikan hasil tampilannya

```
$ prog06.sh [NAMA = pebri]
```

```
$ prog06.sh [NAMA = <CR>] (Terdapat pesan error)
```



```
farid@farid-VirtualBox:~$ . prog06.sh [NAMA = pebri]  
NAMA = pebri
```



```
farid@farid-VirtualBox:~$ .prog06.sh [NAMA = <CR>]  
bash: CR: No such file or directory
```

Analisis

Gambar di atas merupakan perintah instruksi test. Perintah ini merupakan eksekusi dari prog06.sh apabila nama diisi dengan pebri maka seperti pada perintah 1 yaitu Selamat datang pebri, sedangkan pada contoh kedua variabel nama diisi dengan <CR> maka ditampilkan pesan error karena tidak ada di direktori.

4. Modifikasi prog06.sh dengan menggunakan notasi untuk test

```
$ vi prog06.sh
```

```
#!/bin/sh # prog06.sh echo -n
```

```
"NAMA = "Read NAMA if [ "$NAMA" =  
pebri ] then
```

```
echo "Selamat Datang $NAMA" else
```

```
echo "Anda bukan pebri, sorry!" fi
```

```
#!/bin/sh
#prog06.sh
echo -n "NAMA = "
read NAMA
if [ "$NAMA" = amir]
then
echo "Selamat Datang $NAMA"
else
echo "Anda bukan amir,sorry!"
fi
```

Analisis

Gambar di atas merupakan perintah instruksi test. Sama halnya dengan point 2 namun pada modifikasi ini perintah test diganti dengan perintah [], dan akan di eksekusi pada percobaan berikutnya

5. Jalankan program prog06.sh dengan memasukan NAMA = amir

\$. prog06.sh [NAMA= pebri]

```
farid@farid-VirtualBox:~$ . prog06.sh [NAMA =pebri]
NAMA = pebri
bash: [: missing `]'
Anda bukan amir,sorry!
```

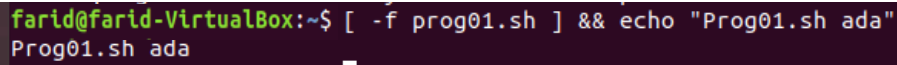
Analisis

Gambar di atas merupakan perintah instruksi test. Meskipun perintah if didefinisikan dengan perintah [] tapi hasil eksekusi prog06.sh akan menampilkan hasil yang sama dengan perintah test sebelumnya.

Percobaan 9 : Notasi && dan ||

1. Bila file prog01.sh ada (TRUE), maka jalankan program berikutnya. File prog01.sh ada, karena itu exit status adalah TRUE, hasil operasi AND masih tergantung pada hasil eksekusi instruksi ke 2, dan dengan demikian instruksi echo akan dijalankan

```
$ [ -f prog01.sh ] && echo "prog01.sh ada"
```



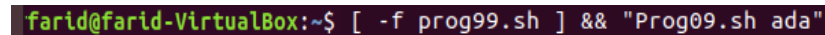
```
farid@farid-VirtualBox:~$ [ -f prog01.sh ] && echo "Prog01.sh ada"
Prog01.sh ada
```

Analisis

Gambar diatas merupakan perintah notasi && dan ||. Bila file prog01.sh ada (TRUE), maka jalankan program berikutnya. File prog01.sh ada, karena itu exit status adalah TRUE, hasil operasi AND masih tergantung pada hasil eksekusi instruksi ke 2, dan dengan demikian instruksi echo akan dijalankan.

2. File prog99.sh tidak ada, karena itu exit status adalah FALSE dan instruksi echo tidak dijalankan

```
$ [ -f prog99.sh ] && echo "prog99.sh ada"
```



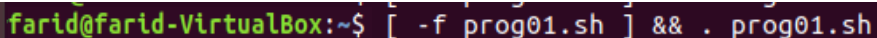
```
farid@farid-VirtualBox:~$ [ -f prog99.sh ] && "Prog09.sh ada"
```

Analisis

Gambar di atas merupakan perintah notasi && dan ||. File prog99.sh tidak ada, karena itu exit status adalah FALSE dan instruksi echo tidak dijalankan.

3. Bila prog01.sh ada maka jalankan shell script tersebut

```
$ [ -f prog01.sh ] && . prog01.sh
```



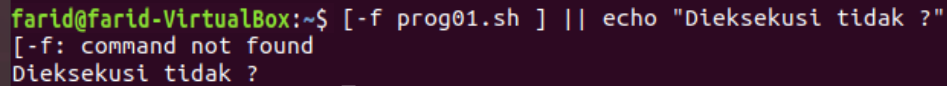
```
farid@farid-VirtualBox:~$ [ -f prog01.sh ] && . prog01.sh
```

Analisis

Gambar di atas merupakan perintah notasi && dan ||. Pada perintah ini adalah untuk menjalankan shell script dari prog01.sh. sama halnya dengan fungsi instruksi dari . prog01.sh.

4. Bila prog01.sh ada maka jalankan program berikutnya. File prog01.sh memang ada, karena itu exit status adalah TRUE, dan karena sudah TRUE maka instruksi echo tidak lagi dijalankan

```
$ [ -f prog01.sh ] || echo "Dieksekusi tidak ?"
```



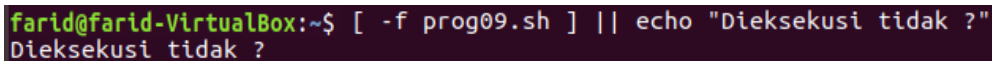
```
farid@farid-VirtualBox:~$ [ -f prog01.sh ] || echo "Dieksekusi tidak ?"
[-f: command not found
Dieksekusi tidak ?
```

Analisis

Gambar di atas merupakan perintah notasi && dan ||. Bila prog01.sh ada maka jalankan program berikutnya. File prog01.sh memang ada, karena itu exit status adalah TRUE, dan karena sudah TRUE maka instruksi echo tidak lagi dijalankan.

5. File prog99.sh tidak ada, karena itu exit status adalah FALSE, hasil masih tergantung atas exit status instruksi ke dua, karena itu instruksi echo dijalankan

```
$ [-f prog99.sh ] || echo "Dieksekusi tidak ?"
```



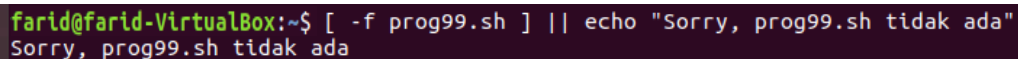
```
farid@farid-VirtualBox:~$ [ -f prog99.sh ] || echo "Dieksekusi tidak ?"
Dieksekusi tidak ?
```

Analisis

Gambar di atas merupakan perintah notasi && dan ||. File prog99.sh tidak ada, karena itu exit status adalah FALSE, hasil masih tergantung atas exit status instruksi ke dua, karena itu instruksi echo dijalankan (dieksekusi).

6. File prog99.sh tidak ada, maka tampilkan pesan error

```
$ [ -f prog99.sh ] || echo "Sorry, prog99.sh tidak ada"
```



```
farid@farid-VirtualBox:~$ [ -f prog99.sh ] || echo "Sorry, prog99.sh tidak ada"
Sorry, prog99.sh tidak ada
```

Analisis

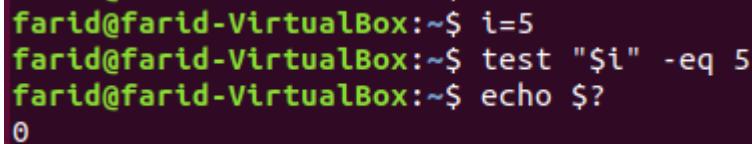
Gambar di atas merupakan perintah notasi && dan ||. File prog99.sh tidak ada, maka tampilkan pesan error seperti perintah di atas.

Percobaan 10 : Operator Bilangan Bulat Untuk Test

1. Menggunakan operator dengan notasi test

```
$ i=5
```

```
$ test "$i" -eq 5 $ echo $?
```



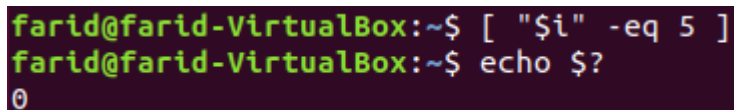
```
farid@farid-VirtualBox:~$ i=5
farid@farid-VirtualBox:~$ test "$i" -eq 5
farid@farid-VirtualBox:~$ echo $?
0
```

Analisis

Gambar di atas merupakan perintah operator bilangan bulat untuk test. Variabel `i` berisi 5 maka dengan notasi test, variabel `i` `-eq` (sama dengan) 5 dan apabila dijalankan instruksi `echo $?` status exit bernilai 0 karena `i` memang berisi dengan nilai 5.

2. Menggunakan operator dengan notasi `[]` (pengganti notasi test)

```
$ [ "$i" -eq 5 ] $ echo $?
```



```
farid@farid-VirtualBox:~$ [ "$i" -eq 5 ]
farid@farid-VirtualBox:~$ echo $?
0
```

Analisis

Gambar di atas merupakan perintah operator bilangan bulat untuk test. Sama halnya dengan yang nomer 1, namun diganti dengan notasi `[]` maka status exit tetap akan bernilai 0.

Percobaan 11 : Operator Logika dan Konstruksi Elif

1. Buatlah file prog07.sh

```
$ vi prog07.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"
```

Analisis

Gambar di atas merupakan perintah operator logika dan konstruksi elif. Perintah

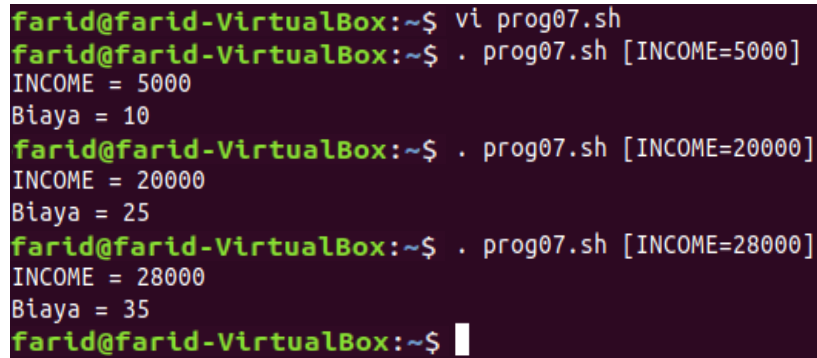
\$ vi prog07.sh berfungsi untuk membuat Shell script prog07.sh yang berisi perintah if dengan variabel income. Ada 3 perintah yaitu BIAYA=10 BIAYA=25 dan BIAYA=35 dan akan dieksekusi pada berikutnya.

2. Jalankan file prog07.sh dan masukan untuk INCOME=5000, 20000, 28000

```
$ . prog07.sh [INCOME=5000 ]
```

```
$ . prog07.sh [INCOME=20000 ]
```

```
$ . prog07.sh [INCOME=28000]
```



```
farid@farid-VirtualBox:~$ vi prog07.sh
farid@farid-VirtualBox:~$ . prog07.sh [INCOME=5000]
INCOME = 5000
Biaya = 10
farid@farid-VirtualBox:~$ . prog07.sh [INCOME=20000]
INCOME = 20000
Biaya = 25
farid@farid-VirtualBox:~$ . prog07.sh [INCOME=28000]
INCOME = 28000
Biaya = 35
farid@farid-VirtualBox:~$
```

Analisis

Gambar di atas merupakan perintah operator logika dan konstruksi elif. Apabila variabel income diisi dengan INCOME antara 0 sampai dengan 10000 maka biaya yang akan ditampilkan adalah 10, sedangkan apabila income berisi nilai antara 10000 sampai dengan 25000 maka biaya yang akan ditampilkan adalah 25 dan jika INCOME berisi lebih dari 25000 maka ditampilkan BIAYA 35 dan tampil seperti gambar di atas.

Percobaan 12 : Hitungan aritmatika

1. Menggunakan utilitas expr

```
$ expr 5 + 1
```

```
$ A=5
```

```
$ expr $A + 2 $ expr $A - 4
```

```
$ expr $A * 2      (Ada pesan error)
```

```
$ expr $A \* 2
```

```
$ expr $A / 6 + 10
```

```
$ expr 17 % 5
```

```
farid@farid-VirtualBox:~$ expr 5 + 1
6
farid@farid-VirtualBox:~$ A=5
farid@farid-VirtualBox:~$ expr $A + 2
7
farid@farid-VirtualBox:~$ expr $A - 4
1
farid@farid-VirtualBox:~$ expr $A * 2
expr: syntax error
farid@farid-VirtualBox:~$ expr $A \* 2
10
farid@farid-VirtualBox:~$ expr $A / 6 + 10
10
farid@farid-VirtualBox:~$
```

Analisis

Gambar di atas merupakan perintah hitungan aritmatika. Untuk perhitungan aritmatika dapat menggunakan utilitas `expr`. Untuk simbol dari penjumlahan adalah `+`, pengurangan `-`, namun untuk perkalian bukan menggunakan perintah `*` tetapi `expr` dari perkalian adalah `*` dan untuk pembagian menggunakan `/` dan `%` digunakan untuk `expr` mod atau sisa hasil bagi dan tampil seperti gambar di atas.

2. Substitusi isi variable denga hasil utilitas `expr`

```
$ A=5
```

```
$ B='expr $A + 1'
```

```
$ echo $B
```

```
farid@farid-VirtualBox:~$ A=5
farid@farid-VirtualBox:~$ B='expr $A + 1'
farid@farid-VirtualBox:~$ echo $B
expr $A + 1
```

Analisis

Gambar di atas merupakan peritnah hitungan aritmatika. Variabel A diisi dengan 5 kemudian variabel 5 berisi substitusi dari perintah `expr $A + 1` maka apabila variabel B dipanggil akan berisi hasil dari aritmatika tersebut.

Percobaan 13 : Instruksi exit

1. Buat shell script prog08.sh

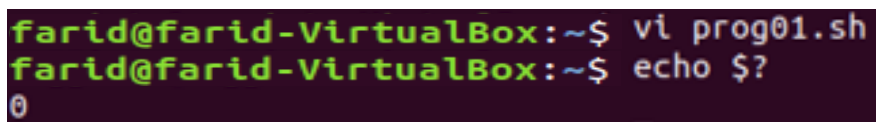
```
$ vi
prog08.sh
#!/bin/sh if
[ -f
prog01.sh ]
then exit 3
else
exit -1 fi
```

Analisis

Gambar di atas merupakan peritntah instruksi exit. Shell script prog08.sh ini berisi instruksi if untuk notasi [] prog01.sh berisi instruksi exit dan akan dieksekusi pada percobaan berikutnya.

2. Jalankan Prog08.sh

```
$ . prog08.sh $ echo $?
```



```
farid@farid-VirtualBox:~$ vi prog01.sh
farid@farid-VirtualBox:~$ echo $?
0
```

Analisis

Gambar di atas merupakan perintah instruksi exit. Apabila prog08.sh dieksekusi dengan . prog08.sh kemudian status exit di tampilkan akan menampilkan nilai 0 dan tampil seperti gambar di atas.

Percobaan 14 : Konstruksi case – esac

1. Buatlah file prog09.sh dengan editor vi

```
$ vi prog09.sh
#!/bin/sh # Prog:

prog09.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 hari ini"
        date
    ;;
    3)
        echo "Kalender bulan ini"
        cal
    ;;
    *)
        echo "Salah pilih !!"
    ;;
esac
```

Analisis

Gambar di atas merupakan perintah Konstruksi Case-Esac. Dengan variabel pemilihan PILIH melalui keyboard. Format penulisan case ini pertama diawali dengan case kemudian variabel pilih diikuti dengan in. Kemudian definisikan untuk nomor 1 hingga 3 setiap nomor ditutup dengan simbol ;;. Untuk pemilihan terakhir apabila memilih selain nomor 1 2 3 digunakan perintah *). Kemudian ditutup dengan esac dan akan dieksekusi pada percobaan berikutnya

2. Jalankan perintah prog09.sh

\$. prog09.sh

```
farid@farid-VirtualBox:~$ vi prog09.sh
farid@farid-VirtualBox:~$ . prog09.sh
1. Siapa yang aktif
2. Tanggal Hari ini
3. Kalender bulan ini
Pilihan: 1
Yang aktif saat ini
farid :0 2019-02-12 18:25 (:0)
farid@farid-VirtualBox:~$ . prog09.sh
1. Siapa yang aktif
2. Tanggal Hari ini
3. Kalender bulan ini
Pilihan: 2
Tanggal hari ini
Sel Mar 19 20:10:00 WIB 2019
farid@farid-VirtualBox:~$ . prog09.sh
1. Siapa yang aktif
2. Tanggal Hari ini
3. Kalender bulan ini
Pilihan: 3
Kalender bulan ini
Maret 2019
Mi Se Se 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
```

Analisis

Gambar di atas merupakan perintah Konstruksi Case-Esac. Apabila memilih nomor 1 maka akan ditampilkan siapa user yang sedang aktif, sedangkan apabila memilih nomor 2 akan ditampilkan tanggal hari ini, apabila memilih nomor 3 akan tampil kalender bulan ini dan apabila memilih selain nomor 1 2 atau 3 maka akan ditampilkan echo Salah pilih !!. Kemudian ditutup dengan esac.

3. Buatlah file prog10.sh yang merupakan bentuk lain dari case

```
#!/bin/sh
#Prog:prog10.sh
echo -n "Jawab (Y/T):"
read JWB
case JWB in
Y | y | ya | YA ) JWB=y ;;
T | t | tidak | Tidak | TIDAK ) JWB=t ;;
esac
```

Analisis

Gambar di atas merupakan perintah Konstruksi Case-Esac. Shell script prog10.sh ini merupakan bentuk lain dari perintah case. Variabel JWB dapat dijawab dengan perintah y/Y/ya/Ya/YA untuk y, sedangkan untuk JWB=t dapat ditulis dengan t/T/tidak/Tidak/TIDAK dan akan dieksekusi pada percobaan berikutnya.

4. Jalankan program prog10.sh

\$. prog10.sh

```
farid@farid-VirtualBox:~$ vi prog10.sh
farid@farid-VirtualBox:~$ . prog10.sh
Jawab (Y/T):y
farid@farid-VirtualBox:~$ . prog10.sh
Jawab (Y/T):t
farid@farid-VirtualBox:~$ . prog10.sh
Jawab (Y/T):ya
farid@farid-VirtualBox:~$ . prog10.sh
Jawab (Y/T):tidak
farid@farid-VirtualBox:~$
```

Analisis

Gambar di atas merupakan perintah Konstruksi Case-Esac. Ini merupakan eksekusi dari prog10.sh. saya menggunakan 3 buah contoh untuk nilai JWB yaitu y, T dan YA.

5. Modifikasi file prog10.sh yang merupakan bentuk lain dari case

```
#!/bin/sh
#Prog:prog10.sh
echo -n "Jawab (Y/T) : \c"
read JWB
case JWB in
    [yY] | [yY] [aA] ) JWB=y ;;
    [tT] | [tT] tidak ) JWB=t ;;
    *) JWB=? ;;
esac
```

Analisis

Gambar di atas merupakan perintah Konstruksi Case-Esac. Selain pada percobaan 14 nomer 3 diatas, perintah case untuk jawaban juga dapat ditulis seperti diatas. Berarti y dapat ditulis huruf kecil atau besar, kemudian untuk “ya” bisa y nya bisa huruf besar/kecil begitupula dengan a nya. Sama halnya dengan ya tidak juga demikian dan akan dieksekusi pada berikutnya.

6. Jalankan program prog10.sh

\$. prog10.sh

```
farid@farid-VirtualBox:~$ vi prog10.sh
farid@farid-VirtualBox:~$ . prog10.sh
Jawab (Y/T) : \cy
bash: prog10.sh: line 6: syntax error near unexpected token `[aA]'
bash: prog10.sh: line 6: `        [yY] | [yY] [aA] ) JWB=y ;;'
farid@farid-VirtualBox:~$ . prog10.sh
Jawab (Y/T) : \ct
bash: prog10.sh: line 6: syntax error near unexpected token `[aA]'
bash: prog10.sh: line 6: `        [yY] | [yY] [aA] ) JWB=y ;;'
farid@farid-VirtualBox:~$
```

Analisis

Gambar di atas merupakan perintah Konstruksi Case-Esac. Ini merupakan eksekusi dari program shell script prog10.sh untuk modifikasi nya dan tampil seperti gambar di atas.

Percobaan 15 : Konstruksi for-do-done

1. Buatlah file prog11.sh

```
#!/bin/sh
#prog : prog11.sh
for NAMA in Farid Aziz
do
    echo "Nama adalah : $NAMA"
done
```

Analisis

Gambar di atas merupakan perintah konstruksi for-do-done. Shell script ini berisi konstruksi for-do-done dimana for untuk variabel nama ada pebri messi neymar suarez. Kemudian dijalankan dengan perintah do dan ditutup dengan done dan akan dieksekusi pada percobaan berikutnya.

2. Jalankan program prog11.sh

\$. prog11.sh

```
farid@farid-VirtualBox:~$ vi prog11.sh
farid@farid-VirtualBox:~$ . prog11.sh
Nama adalah : Farid
Nama adalah : Aziz
farid@farid-VirtualBox:~$
```

Analisis

Gambar di atas merupakan perintah konstruksi for-do-done. Apabila prog11.sh ini dijalankan perintah echo akan ditampilkan untuk setiap NAMA karena menggunakan pengulangan for dan tampil seperti tampilan di atas.

3. Buatlah file prog12.sh yang berisi konstruksi for dan wildcard

```
#!/bin/sh
#prog : prog12.sh
for F in*
do
    echo $F
done
~
```

Analisis

Gambar di atas merupakan perintah konstruksi for-do-done. Shell script prog12.sh ini berisi konstruksi for dan wild card. Program ini akan menampilkan nama file yang berada pada current directory dan akan dieksekusi pada percobaan berikutnya

4. Jalankan program prog12.sh

\$. prog12.sh

```
farid@farid-VirtualBox:~$ vi prog12.sh
farid@farid-VirtualBox:~$ . prog12.sh
bash: prog12.sh: line 3: syntax error near unexpected token
bash: prog12.sh: line 3: `for F in*'
farid@farid-VirtualBox:~$
```

Analisis

Gambar di atas merupakan perintah konstruksi for-do-done. Apabila prog12.sh ini dijalankan maka seluruh nama directory pada current directory akan ditampilkan dan tampil seperti gambar.

5. Modifikasi file prog12.sh.

```
#!/bin/sh
#prog : prog12.sh
for F in *.lst
do
    echo -l $F
done
~
```

Analisis

Gambar di atas merupakan perintah yaitu konstruksi for-do-done. Modifikasi prog12.sh ini akan menampilkan long list dari file yang mempunyai ekstensi file .lst dan akan dieksekusi pada percobaan berikutnya

6. Jalankan prog12.sh

\$. prog12.sh

```
farid@farid-VirtualBox:~$ vi prog12.sh
farid@farid-VirtualBox:~$ . prog12.sh
-l *.lst
farid@farid-VirtualBox:~$
```

Analisis

Gambar di atas merupakan perintah konstruksi for-do-done. Tidak ditemukan file/direktori dengan ekstensi .lst pada current directory ini sehingga ditampilkan pesan error dan tampil seperti gambar.

Percobaan 16 : Konstruksi while-do-done

1. Buatlah program 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 disini setelah break"
~
```

Analisis

Gambar di atas merupakan perintah yaitu konstruksi while-do-done. Program shell script untuk prog13.sh ini berisi perintah pengulangan serta terdapat 4 pilihan. Variabelnya adalah PILIH –ne 4 dan akan dieksekusi pada percobaan berikutnya.

2. Jalankan prog13.sh

\$. prog13.sh

```
farid@farid-VirtualBox:~$ vi prog13.sh
farid@farid-VirtualBox:~$ . prog13.sh
1. Siapa yang aktif
2. Tanggal hari ini
3. Kalender bulan ini
4. Keluar
Pilihan : \c
1

1. Siapa yang aktif
2. Tanggal hari ini
3. Kalender bulan ini
4. Keluar
Pilihan : \c
4
Program berlanjut disini setelah break
farid@farid-VirtualBox:~$
```

Analisis

Gambar di atas merupakan perintah konstruksi while-do-done. Ketika prog13.sh ini dijalankan apabila memilih nomor 1 2 atau 3 maka akan diulangi atau ditampilkan lagi pilihan tersebut karena perintahnya tidak berisi. Sedangkan apabila memilih nomor 4 maka pengulangan akan berhenti kemudian tampil tulisan dan tampil seperti gambar.

Percobaan 17 : Instruksi dummy

1. Modifikasi file prog13.sh, kemudian jalankan!

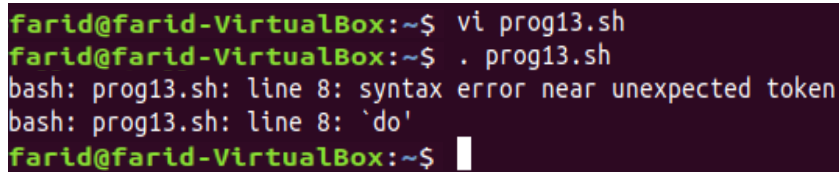
```
#!/bin/sh
#Prog : prog13.sh
PILIH=1
while
do
echo "1. Siapa yang aktif"
echo "2. Tanggal hari ini"
echo "3. Kalender bulan ini"
echo "4. Keluar"
echo "Pilihan : \c"
read PILIH
if [ $PILIH -eq 4 ]
then
break
fi
clear
done
echo "Program berlanjut disini setelah break"
```

Analisis

Gambar di atas merupakan perintah yaitu instruksi dummy. Modifikasi ini terlihat pada penulisan while : sebelumnya ditambahkan dengan notasi [\$PILIH -ne 4] sedangkan pada prog14.sh sekarang tidak ada dan akan dieksekusi pada percobaan berikutnya

2. Jalankan prog13.sh

\$. prog13.sh

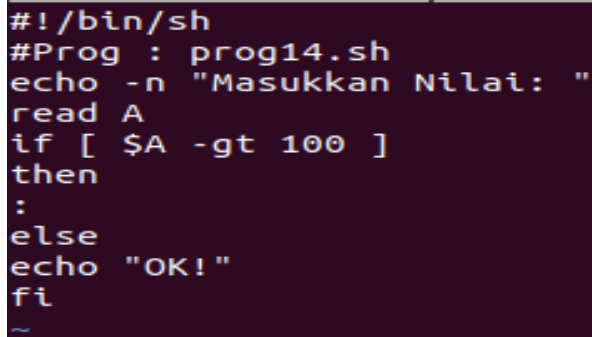


```
farid@farid-VirtualBox:~$ vi prog13.sh
farid@farid-VirtualBox:~$ . prog13.sh
bash: prog13.sh: line 8: syntax error near unexpected token
bash: prog13.sh: line 8: `do'
farid@farid-VirtualBox:~$
```

Analisis

Gambar di atas merupakan perintah instruksi dummy. Hasil dari eksekusi modifikasi prog13.sh ini akan sama dengan perintah sebelumnya yaitu pada percobaan 17 dan tampil seperti gambar.

3. Buatlah file prog14.sh yang berisi instruksi dummy untuk konstruksi if, kemudian jalankan!



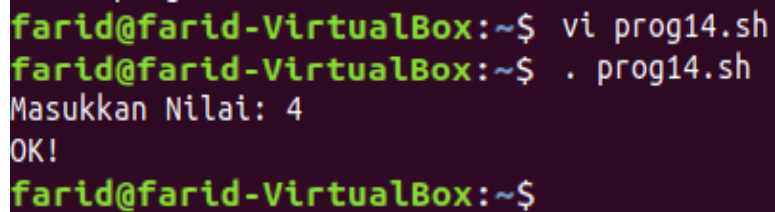
```
#!/bin/sh
#Prog : prog14.sh
echo -n "Masukkan Nilai: "
read A
if [ $A -gt 100 ]
then
:
else
echo "OK!"
fi
~
```

Analisis

Gambar di atas merupakan perintah instruksi dummy. Masih pada shell script, pada program prog14.sh ini berisi instruksi dummy untuk konstruksi if. Variabel A dinyatakan dengan nilai -ne 100. Then echo OK dan akan dieksekusi pada percobaan berikutnya

4. Jalankan prog14.sh

\$. prog14.sh



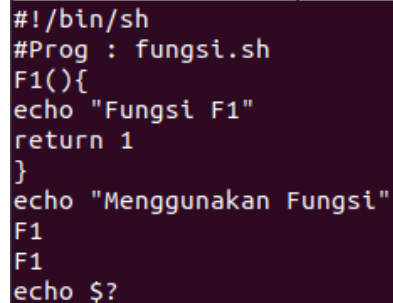
```
farid@farid-VirtualBox:~$ vi prog14.sh
farid@farid-VirtualBox:~$ . prog14.sh
Masukkan Nilai: 4
OK!
farid@farid-VirtualBox:~$
```

Analisis

Gambar di atas merupakan perintah instruksi dummy. Apabila dijalankan jika nilai yang dimasukan masih kurang dari 100 maka ditampilkan OK !, sedangkan apabila nilai lebih dari 100 maka tidak akan ada pesan OK ! dan tampil seperti gambar.

Percobaan 18 : Fungsi

1. Buatlah file fungsi.sh, kemudian jalankan!



```
#!/bin/sh
#Prog : fungsi.sh
F1(){
echo "Fungsi F1"
return 1
}
echo "Menggunakan Fungsi"
F1
F1
echo $?
```

Analisis

Gambar di atas merupakan perintah fungsi. File fungsi.sh pada shell script ini berisi perintah fungsi untuk return 1 dan status exit setelah menjalankan fungsi dan akan dieksekusi pada percobaan berikutnya

2. Jalankan fungsi.sh

\$. fungsi.sh

```
farid@farid-VirtualBox:~$ vi fungsi.sh
farid@farid-VirtualBox:~$ . fungsi.sh
Menggunakan Fungsi
Fungsi F1
Fungsi F1
1
farid@farid-VirtualBox:~$
```

Analisis

Gambar di atas merupakan perintah fungsi. File fungsi.sh ini apabila dijalankan akan berstatus exit 1 karena bernilai false dan tampil seperti gambar.

3. Menggunakan variable pada fungsi dengan memodifikasi file **fungsi.sh**, kemudian jalankan!

```
#!/bin/sh
#Prog : fungsi.sh
F1()
{
Honor=10000
echo "Fungsi F1"
return 1
}
echo "Menggunakan Fungsi"
F1
F1
echo "Nilai baik saya adalah $?"
echo "Honor = $Honor"
```

Analisis

Gambar di atas merupakan perintah fungsi. Modifikasi file fungsi.sh ini hanya ditambah variabel Honor dengan 10000 dan echo Nilai balik adalah \$? dan akan dieksekusi pada percobaan berikutnya

4. Jalankan fungsi.sh

\$. fungsi.sh

```
farid@farid-VirtualBox:~$ vi fungsi.sh
farid@farid-VirtualBox:~$ . fungsi.sh
Menggunakan Fungsi
Fungsi F1
Fungsi F1
Nilai baik saya adalah 1
Honor = 10000
farid@farid-VirtualBox:~$
```

Analisis

Gambar di atas merupakan perintah yaitu fungsi. Tidak ada perbedaan fungsi dalam file.sh hasil modifikasi ini hanya penambahan nilai dari variabel Honor saja dan tampil seperti gambar.

5. Menggunakan variable pada fungsi dengan memodifikasi file **fungsi.sh**, kemudian jalankan!

```
#!/bin/sh
#Prog : fungsi.sh
F1()
{
    local Honor=10000
    echo "Fungsi F1"
    return 1
}
echo "Menggunakan Fungsi"
F1
F1
echo "Nilai baik saya adalah $?"
echo "Honor = $Honor"
~
```

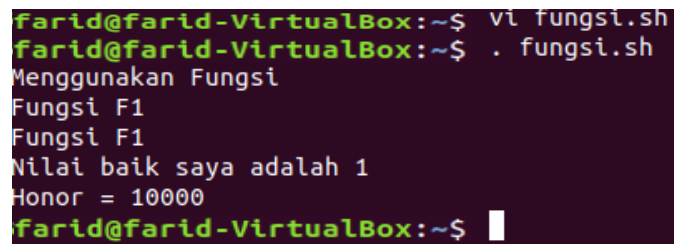
Analisis

Gambar di atas merupakan perintah yaitu fungsi. Percobaan ini juga masih merupakan modifikasi dari file fungsi.sh dengan penambahan local sebelum

pendefinisian variabel Honor=10000 dan akan dieksekusi pada percobaan berikutnya

6. Jalankan fungsi.sh

\$. fungsi.sh



```
farid@farid-VirtualBox:~$ vi fungsi.sh
farid@farid-VirtualBox:~$ . fungsi.sh
Menggunakan Fungsi
Fungsi F1
Fungsi F1
Nilai baik saya adalah 1
Honor = 10000
farid@farid-VirtualBox:~$
```

Analisis

Gambar di atas merupakan perintah fungsi. Dan untuk hasilnya pun tidak ada perubahan dari modifikasi file fungsi.sh sebelumnya. Sama saja dengan percobaan 18 nomer 3 dan tampil seperti gambar.

LATIHAN

1. Buatlah program salin.sh yang menyalin file (copy) sebagai berikut : Salin.sh file –asal file-tujuan

Dengan ketentuan :

- Bila file asal tidak ada, berikan pesan, salin gagal.
- Bila file tujuan ada tersebut directory, beri pesan bahwa file tidak bisa disalin ke direktori
- Bila file tujuan ada dan file biasa, beri pesan apakah file tersebut akan dihapus, bila dijawab dengan “Y”, maka copy file tersebut
- Bila file tujuan belum ada, lakukan copy

Untuk mengambil nama file, gunakan parameter \$1 dan \$2. Bila jumlah parameter tidak sama (\$#) dengan 2, maka beri pesan exit=-1

```
#!/bin/sh
#file : salin
#usage : salin.sh fasal ftujuan
if [ $# -ne 2 ]
then
    echo "Error, usage : salin.sh file-asal file-tujuan"
    exit -1
fi
fasal=$1
ftujuan=$2
echo "salin.sh $fasal $ftujuan"
if [ $2 = -d ]
then
    echo -n "copy file? (Y/T) : "
    read jwb
    case $jwb in
        y | Y | ya | Ya | YA ) jwb=y;;
        t | T | tidak | Tidak | ) jwb=t;;
        *) jwb="";;
    esac
else
    exit -1
fi
```

Software Updater

Analisis

Gambar di atas merupakan program copy file, dimana pada gambar di atas jika file tidak berhasil ditemukan maka nantinya akan langsung di close. Untuk pilihan digunakan perintah if else then, tetapi karena perintah pertama pada shell script salin.sh menyatakan jika file tidak ada maka akan di close, dan memang si file ini tidak ada, sehingga perintah yang dikerjakannya pada saat shell script dieksekusi menggunakan perintah . Salin.sh adalah perintah pilihan nomor 1 yaitu “keluar” karena tidak ada filenya dan tampil seperti gambar di atas.

2. Buat program yang memeriksa nama direktori, jika parameter tersebut adalah direktori, maka jalankan instruksi `ls -ld` pada direktori tersebut. Namakan program tersebut **checkdir.sh**. gunakan notasi `[-d namadirektori]` dan pilih logical `&&` atau `||` pada level shell

```
#!/bin/sh
#file : checkdir.sh
#usage : checkdir.sh mydir
#
if [ $# -ne 1 ]
then
    echo "Error, usagge : checkdir.sh DirectoryName"
    exit 1
fi
[ -d mydir ] && echo "mydir ada"
```

Analisis

Gambar di atas merupakan program memeriksa nama direktori. Seperti yang terjadi pada latihan no. 1 tadi, ketika dijalankan program `checkdir.sh` ini tidak menemukan directory yang di cari sehingga secara otomatis akan di “closed” atau exit dan tampil seperti gambar di atas.

3. Dengan shell script `pph.sh`, hitung PPH per tahun dengan ketentuan sebagai berikut :

- 10 juta pertama PPH 15%,
- 25 juta berikutnya (sisanya) PPH 25%
- Bila masih ada sisa, maka sisa tersebut PPH 35%.

Contoh :

Gaji 8 juta

PPH = 15% * 8 juta

Gaji 12 juta

PPH = 15% * 10 juta + 25% (12-10) juta

Gaji 60 juta

PPH = 15% * 10 juta + 25% * 25 juta + 35% * (60-10-25) juta

Debugging : untuk melakukan tracing (debug) gunakan opsi `-x` pada eksekusi shell.

```
#!/bin/sh
#prog : pph.sh
#
echo -n 'Berikan gaji dalam rupiah'
read gaji
pkp=10000
 '[' 20000 -le 10000 ']'
expr 20000 - 10000
gaji=10000
pph=1500
pkp=25000
 '[' 10000 -le 25000 ']'
pkp=10000
expr 1500 + 10000 '*' 25/ 100
pph=4000
echo 'Pajak Penghasilan = 4000'
```

```
farid@farid-VirtualBox:~$ vi pph.sh
farid@farid-VirtualBox:~$ sh -x pph.sh
+ echo -n Berikan gaji dalam rupiah
Berikan gaji dalam rupiah+ read gaji
20000
+ pkp=10000
+ [ 20000 -le 10000 ]
+ expr 20000 - 10000
10000
+ gaji=10000
+ pph=1500
+ pkp=25000
+ [ 10000 -le 25000 ]
pph.sh: 12: pph.sh: [10000: not found
+ pkp=10000
+ expr 1500 + 10000 * 25/ 100
expr: non-integer argument
+ pph=4000
+ echo Pajak Penghasilan = 4000
pajak penghasilan = 4000
farid@farid-VirtualBox:~$
```

Analisis

Gambar di atas merupakan shell script pph.sh, menghitung PPH per tahun dengan ketentuan 10 juta pertama PPH 15%, 25 juta berikutnya (sisa) PPH 25%, bila masih ada sisa, maka sisa tersebut PPH 35%. Untuk membuat sebuah file pph.sh gunakan perintah vi pph.sh. masuk ke shell dan buat sebuah perintah dalam vi

pph.sh dengan memasukan perintah yang ada pada gambar di atas. Gunaka tombol Esc ':' dan wq untuk keluar dari shell dan menyimpan. Kemudian jalankan pph.sh dengan perintah bash pph.sh dan untuk debugging untuk melakukan tracing (debug) gunakan opsi -x pada eksekusi shell, maka hasilnya tampil seperti gambar di atas.

4. Buatlah program **myprog.sh** yang memproses parameter \$1, nilai parameter harus berupa string :

start stop status

restart reload

Bila buka dari string tersebut, maka berikan pesan error. Sempurnakan program dibawah ini!

```
#!/bin/sh
#
case "$1" in
    start)
        echo "ini adalah start"
        ;;
    stop)
        echo "ini adalah stop"
        ;;
    *)
        echo "$usage:$0 {start|stop|restart|reload|status}"
        ;;
esac
return
```

```
farid@farid-VirtualBox:~$ vi myprog.sh
farid@farid-VirtualBox:~$ . myprog.sh
usage:bash {start|stop|restart|reload|status}
farid@farid-VirtualBox:~$ . myprog.sh start
ini adalah start
farid@farid-VirtualBox:~$ . myprog.sh stop
ini adalah stop
farid@farid-VirtualBox:~$ . myprog.sh restart
usage:bash {start|stop|restart|reload|status}
farid@farid-VirtualBox:~$ chmod +x myprog.sh
farid@farid-VirtualBox:~$ ./myprog.sh
$usage:./myprog.sh {start|stop|restart|reload|status}
farid@farid-VirtualBox:~$
```

Analisis

Gambar di atas merupakan tahapan membuat program **myprog.sh** yang memproses parameter \$1, nilai parameter harus berupa string **start**, **stop**, **status**, **restart**, dan **reload**. Isi dari file myprog.sh dengan perintah untuk melakukan seleksi terhadap inputan yang dimasukan. Jika inputan yang dimasukan start, stop, status, restart, atau reload maka akan muncul output seperti gambar di atas dan jika tidak, akan muncul informasi kesalahan dan tampil seperti gambar di atas.

5. Buat sebuah fungsi pada script **confirm.sh** yang memberikan konfirmasi jawaban Yes, No atau Continue. Jika jawaban Yes, maka beri nilai balik 0, No = 1 dan continue = 2. Modifikasi program!

```
#!/bin/sh
#confirm wheter we really wan to run this service
confirm() {
local YES="Y"
local NO="N"
locat CONT="C"
while :
do
echo -n "(Y)es/(N)o/(C)ontinue? {Y} "
read answer
answer='echo "$answer" | tr '[a-z]' '[A-Z]''
if [ "$answer" = "" -o "$answer" = $YES ]
then
return 0
elif [ $ -gt YES -a $jwb = $YES ]
then
return 2
elif [ $ -gt NO -a $jwb = $NO ]
then
return 1
fi
done
}
```

```
if [ $? -eq 0 ]
then
echo "Jawaban YES OK"
elif [ $? -eq 1 ]
then
echo "Jawaban NO"
else
echo "Jawaban CONTINUE"
fi
```

```
farid@farid-VirtualBox:~$ vi confirm.sh
farid@farid-VirtualBox:~$ . confirm.sh
farid@farid-VirtualBox:~$ vi testp.sh
farid@farid-VirtualBox:~$ . testp.sh
Jawaban YES OK
farid@farid-VirtualBox:~$
```


Analisis

Gambar di atas merupakan latihan nomer 5 yaitu membuat sebuah fungsi pada script **confirm.sh** yang memberikan konfirmasi jawaban Yes, No atau Continue. Jika jawaban Yes, maka beri nilai balik 0, No = 1 dan continue = 2 serta memodifikasi program tersebut. Pada perintah yang terdapat di modul tidak bisa dijalankan, sehingga digunakan modifikasi terhadap isi perintah. Lalu digunakan perintah elif seperti yang terlihat pada gambar di atas, dan ketika dijalankan hasilnya akan tampil seperti di atas dengan output tulisan “Jawaban YES OK”.