Praktikum 6-B

Pemrograman Shell

POKOK BAHASAN:

- ✓ Pemrograman Shell

TUJUAN BELAJAR:

Setelah mempelajari materi dalam bab ini, mahasiswa diharapkan mampu:
- ✓ Menggunakan struktur case – esac.
- ✓ Loop dengan while, for, do while.
- ✓ Membuat fungsi dan mengetahui cara memanggil fungsi tersebut.

DASAR TEORI:

1 INSTRUKSI TEST

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

- 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 directory

  Test dapat dilakukan untuk memeriksa apakah file ada (Exist), dapat dibaca, dapat ditulis, kosong dan lainnya.

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 ]
```

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

2 LOGICAL && DAN || (SHELL LEVEL)

Notasi && dan || digunakan untuk menggabungkan instruksi shell sebagai alternatif untuk if then else. Notasi && dan || sering ditemukan dalah 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 karena itu, `instruksi2` tidak dijalankan. Sebaliknya bila hasil `instruksi1` adalah TRUE(0), maka `instruksi2` dijalankan

- `instruksi1 || instruksi2`

  shell akan mengeksekusi `instruksi1`, 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.

3 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

4 OPERATOR LOGICAL (TEST LEVEL)

Logical operator terdiri dari AND, OR dan NOT. Operator ini menggabungkan hasil ekspresi sebagai berikut :

NOT : symbol **!**

	!
True	False
False	True

AND : symbol **-a**

V1	V2	V1 –a V2
False	False	False
False	True	False
True	False	False

True	True	True

OR : symbol **-o**

V1	V2	V1 –o V2
False	False	False
False	True	True
True	False	True
True	True	True

5 KONSTRUKSI IF THEN ELSE IF

```
if instruksi1
then
     instruksi1.1
     instruksi1.2
     ……
elif instruksi2
then

     instruksi2.1
     instruksi2.2
     ……
else
     instruksi3.1
     instruksi3.2
     ……
fi
```

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

6 HITUNGAN ARITMETIKA

Tipe dari variable SHELL hanya satu yaitu STRING. Tidak ada tipe lain seperti Numerik, Floating, Boolean atau lainnya. Akibatnya variable ini tidak dapat membuat perhitungan aritmetika, misalnya :

```
A=5
B=$A +1    ## error
```

UNIX menyediakan utilitas yang bernama **expr** yaitu suatu utilitas yang melakukan aritmetika sederhana.

7 INSTRUKSI EXIT

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

8 KONSTRUKSI CASE

Case digunakan untuk menyederhanakan pemakaian if yang berantai, sehingga dengan case, kondisi dapat dikelompokkan secara logis dengan 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 denan esac dan pada setiap kelompok instruksi diakhiri dengan ;;. Pada akhir pilihan yaitu *) yang berarti adalah “default”, bila kondisi tidak memenuhi pola sebelumnya

9 KONSTRUKSI FOR

For digunakan untuk pengulangan dengan menggunakan var yang pada setiap pengulangan akan diganti dengan nilai yang berada pada daftar (list).

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

10 KONSTRUKSI WHILE

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

```
while kondisi
do
      instruksi1
      instruksi2
      ……
done
```

11 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).

Simbol instruksi dummy adalah $\Rightarrow$:

12 FUNGSI

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 didefinisikan dalam fungsi sebagai variable local atau global. Hal yang perlu diperhatikan, nama variable yang digunakan dalam sebuah fungsi, jangan sampai bentrok dengan nama variable yang sam adi luar fungsi, sehingga tidak terjadi isi variable berubah.

TUGAS PENDAHULUAN :

Sebagai tugas pendahuluan, bacalah dasar teori diatas kemudian buatlah program Shell untuk Latihan 1 sampai dengan 5.

Percobaan 8 : Instruksi Test

1. Menggunakan instruksi test, perhatikan spasi antara

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

2. Aplikasi test dengan konstruksi if

```
$ vi prog06.sh
#!/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
```

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

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

4. Modifikasi `prog06.sh` dengan menggunakan notasi untuk test

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

5. Jalankan program `prog06.sh` dengan memasukkan NAMA = amir

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

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"
   ```

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

   ```
   $ [ -f prog99.sh ] && echo "Prog99.sh ada"
   ```

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

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

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

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

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

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

Percobaan 10 : Operator bilangan bulat untuk test

1. Menggunakan operator dengan notasi test

   ```
   $ i=5
   $ test "$i" –eq 5
   $ echo $?
   ```

2. Menggunakan operator dengan notasi [ ] (penganti notasi test)

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

Percobaan 11 : Operator Logical 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”
```

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

```
$ . prog07.sh [INCOME=5000]
$ . prog07.sh [INCOME=20000]
$ . prog07.sh [INCOME=28000]
```

Percobaan 12 : Hitungan aritmetika

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
```

2. Substitusi isi variable dengan hasil utilitas expr

```
$ A=5
$ B=`expr $A + 1`
$ echo $B
```

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
```

2. Jalankan script `prog08.sh` dan periksa status exit

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

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
```

2. Jalankan program prog09.sh, cobalah beberapa kali dengan inputan yang berbeda

```
$ . prog09.sh
```

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

```
$ vi prog10.sh
#!/bin/sh
# Prog: prog10.sh

echo –n “Jawab (Y/T) : “
read JWB

case $JWB in
y | Y | ya |Ya |YA ) JWB=y ;;
t | T | tidak | Tidak | TIDAK ) JWB=t ;;
esac
```

4. Jalankan program prog10.sh, cobalah beberapa kali dengan inputan yang berbeda

```
$ . prog10.sh
```

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

```
$ vi prog10.sh
#!/bin/sh
# Prog: prog10.sh

echo –n “Jawab (Y/T) : \c“
read JWB

case $JWB in
[yY] | [yY][aA] ) JWB=y ;;
[tT] | [tT]idak ) JWB=t ;;
                *) JWB=? ;;
esac
```

6. Jalankan program prog10.sh, cobalah beberapa kali dengan inputan yang berbeda

```
$ . prog10.sh
```

Percobaan 15 : Konstruksi for-do-done

1. Buatlah file `prog11.sh`

```
$ vi prog11.sh
#!/bin/sh
# Prog: prog11.sh

for NAMA in bambang harry kadir amir
do
        echo "Nama adalah : $NAMA"
done
```

2. Jalankan program `prog11.sh`

```
$ . prog11.sh
```

3. Buatlah file `prog12.sh` yang berisi konstruksi for dan wildcard, program ini akan menampilkan nama file yang berada di current direktori

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

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

4. Jalankan program `prog12.sh`

```
$ . prog12.sh
```

5. Modifikasi file `prog12.sh`, program ini akan menampilkan long list dari file yang mempunyai ekstensi lst

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

for F in *.lst
do
        ls –l $F
done
```

6. Jalankan program `prog12.sh`

```
$ . prog12.sh
```

Percobaan 16 : Konstruksi while-do-done

1. Buatlah file `prog13.sh`

```
$ vi 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”
```

2. Jalankan program `prog13.sh`

```
$ . prog13.sh
```

Percobaan 17 : Instruksi dummy

1. Modifikasi file `prog13.sh`

```
$ vi prog13.sh
#!/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 di sini setelah break”
```

2. Jalankan program `prog13.sh`

```
$ . prog13.sh
```

3. Buatlah file `prog14.sh` yang berisi instruksi dummy untuk konstruksi if

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

echo –n “Masukkan nilai : “
read A
if [ $A –gt 100 ]
then
        :
else
        echo “OK !”
fi
```

4. Jalankan program `prog14.sh` beberapa kali dengan input yang berbeda

```
$ . prog14.sh
```

Percobaan 18 : Fungsi

1. Buatlah file `fungsi.sh`

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

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

2. Jalankan program `fungsi.sh`

```
$ . fungsi.sh
```

3. Menggunakan variable pada fungsi dengan memodifikasi file `fungsi.sh`

```
$ vi 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"
```

4. Jalankan program `fungsi.sh`

```
$ . fungsi.sh
```

5. Menggunakan variable pada fungsi dengan memodifikasi file `fungsi.sh`

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

F1( )
{
        local Honor=10000
        echo "Fungsi F1"
        return 1
}

echo "Menggunakan Fungsi"
F1
F1
echo "Nilai balik adalah $?"
echo "Honor = $Honor"
```

6. Jalankan program `fungsi.sh`

```
$ . fungsi.sh
```

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 dan file tersebut adalah directory, beri pesan bahwa file tida k bisa disalin ke direktori
   - Bila file tujuan ada dan file biasa, beri pesan apakan 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.sh
# 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"
……
……
```

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 logic al && atau || pada level shell.

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

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

   - 10 juta pertama PPH 15%
   - 25 juta berikutnya (sisa) 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 + 25% * (60-10-25) juta

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

```
$ sh –x pph.sh
+ echo –n 'Berikan gaji dalam ribuan rupiah : '
Berikan gaji dalam ribuan rupiah : + read gaji
20000
```

```
+ 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'
Pajak Penghasilan = 4000
```

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 di bawah ini untuk keperluan tersebut

```
#!/bin/sh
# See how we were called
case "$1" in
  start)
        echo "Ini adalah start"
        ;;
  stop)
        echo "Ini adalah stop"
        ;;
  *)
  echo $"Usage:$0 {start|stop|restart|reload|status}"
        ;;
esac
return
```

5. Buat sebuah fungsi pada script ***confirm.sh*** yang memberikan konfirmasi jawaban **Y**es, **N**o atau **C**ontinue. Jika jawaban Yes, maka beri nilai balik 0, No = 1 dan Continue = 2.

Modifikasi kerangka program berikut untuk memenuhi permintaan tersebut.

```
#!/bin/sh
# Confirm whether we really want to run this service
confirm() {
  local YES="Y"
  local NO="N"
  local CONT="C"
```

```
  while :
  do
        echo –n "(Y)es/(N)o/(C)ontinue? {Y] "
        read answer
        answer=`echo "$answer" | tr '[a-z]' '[A-Z]'`

        if [ "$answer" = "" –0 "$answer" = $YES ]
        then
             return 0
        elif ….
        then
             return 2
        elif ….
        then
             return 1
        fi
  done
}
```

Test fungsi diatas dengan program berikut :

```
$ vi testp.sh
. confirm.sh
confirm
if [ $? –eq 0 ]
then
  echo "Jawaban YES OK"
elif [ $? =eq 1 ]
then
  echo "Jawaban NO"
else
  echo "Jawaban CONTINUE"
fi
```

Perhatikan baris pertama, adalah loading dari fungsi confirm yang terdapat di script confirm.sh. Setelah eksekusi script tersebut, maka fungsi confirm dapat digunakan.

LAPORAN RESMI:

1. Analisa hasil percobaan yang Anda lakukan.
2. Kerjakan latihan diatas dan analisa hasil tampilannya.
3. Berikan kesimpulan dari praktikum ini.