Praktikum 6-A

Pemrograman Shell

POKOK BAHASAN:

- ✓ Pemrograman Shell

TUJUAN BELAJAR:

Setelah mempelajari materi dalam bab ini, mahasiswa diharapkan mampu:

- ✓ Mempelajari elemen dasar shell script
- ✓ Membuat program shell interaktif
- ✓ Menggunakan parameter dalam program
- ✓ Mempelajari test kondisi serta operator logic yang terkait dengan instruksi test
- ✓ Mengenal variable built-in dari shell
- ✓ Membuat aplikasi dengan shell menggunakan konstruksi if-then-else
- ✓ Menggunakan struktur case – esac.
- ✓ Loop dengan while, for, do while.
- ✓ Membuat fungsi dan mengetahui cara memanggil fungsi tersebut.

DASAR TEORI:

1 SHELL SCRIPT

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

```
#!/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 di situ. 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 di antara nama variable dan konstanta

2 VARIABEL

Variable shell adalah variable yang dapat mempunyai nilai berupa nilai String. Tata penulisan variable adalah sebagai berikut :

nama_var = *nilai_var*

Variable harus dimulai dengan alfabet, disusul dengan alfanumerik dan karakter lain. Variabel dapat ditulis dalam huruf kecil atau huruf besar atau campuran keduanya. Shell membedakan huruf besar dan huruf kecil (*case sensitive*), contoh :

```
VPT=poltek
i=5
```

Pemberian nilai variable tidak boleh dipisahkan dengan spasi, karena shell akan menganggap pemisahan tersebut sebagai parameter, contoh :

```
VPT   =poltek          ##error
VPT=   poltek          ##error
```

Untuk melihat nilai/isi dari sebuah variable, gunakan tanda $ di depan nama variable tersebut. Pada shell, instruksi echo dapat menampilkan isi variable tersebut, contoh :

```
VPT=poltek
echo $VPT

Gaji=450000
echo $Gaji
echo $VPT $Gaji
```

Bila menggunakan string yang terdiri dari lebih dari satu kata, maka string tersebut harus berada dalam tanda kutip atau apostrof, contoh :

```
VPT=poltek
VPT2=”poltek elektronika ITS”
```

3 MEMBACA KEYBOARD

Nilai variable dapat diisi melalui keyboard (stdin) dengan instruksi `read`.

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 dan $9. Nama program shell (nama script) direpresentasikan melalui variable $0.

Jumlah parameter dinyatakan sebagai $#. Bila tidak memberikan parameter, maka nilai $# adalah 0.

Shell variable $* menyatakan seluruh string yang menjadi parameter / argumen sebuah script ($@ mempunyai arti yang 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 spesial $?. Indikasi yang diberikan adalah :

- Bila program berakhir dengan sukses, $? = 0
- Bila program berakhir dengan error, $? ≠ 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 selanjutnyua masuk ke dalam blok `then`. Bila tidak 0, maka alur program diteruskan setelah kunci kata `fi`.

7 KONSTRUKSI IF THEN ELSE

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

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

8 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.

9 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.

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, 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

12 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.

13 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.

14 INSTRUKSI EXIT

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

15 KONSTRUKSI CASE

Case digunakan untuk menyederhanakan pemakaian if yang berantai, sehingga dengan case, kondisi dapat dikelompokkan secara lo gis 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

16 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
```

17 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
```

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).

Simbol instruksi dummy adalah ⇒ :

19 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 :

1. Login sebagai user.
2. Bukalah Console Terminal dan lakukan percobaan-percobaan di bawah ini kemudian analisa hasil percobaan.
3. Selesaikan soal-soal latihan.

Percobaan 1 : Membuat shell script

1. Buatlah file prog01.sh dengan editor vi

```
$ vi prog01.sh
#!/bin/sh
# Program shell
#
var1=x
var2=8
```

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

```
$ . prog01.sh
```

3. Untuk menjalankan shell, dapat juga dengan membuat executable file dan dieksekusi relatif dari current directory

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

Percobaan 2 : Variabel

1. Contoh menggunakan variable pada shell interaktif

```
$ VPT=poltek
$ echo $VPT
```

2. Pemisahan 2 kata dengan spasi menandakan eksekusi 2 buah instruksi. 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
$ echo $VPT2
```

3. Menggabungkan dua variable atau lebih

```
$ V1=poltek
$ V2=':'
$ V3=elektronika
$ V4=$V1$V2$V3
$ echo $V4
```

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
```

5. Variabel dapat berisi instruksi, yang kemudian bila dijadikan input untuk shell, instruksi tersebut akan dieksekusi

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

6. Modifikasi file `prog01.sh` berikut

```
$ vi prog01.sh
#!/bin/sh
V1=poltek
V2=':'
V3=elektronika
echo "Pemrograman shell"
echo $V1$V2$V3
V3=ITS
echo $V1$V2 di $V3
```

7. Cara sederhana mengeksekusi shell adalah dengan menggunakan notasi titik di depan nama shell script tersebut. Bila direktori actual tidak terdaftar dalam PATH, maka command tersebut tidak dapat ditemukan. Bila script belum executable, script tidak dapat dieksekusi.

```
$ . prog01.sh
$ prog01.sh          (Terdapat pesan error)
$ ./prog01.sh        (Terdapat pesan error)
$ chmod +x prog01.sh
$ ./prog01.sh
```

Percobaan 3 : Membaca keyboard

1. Menggunakan instruksi `read`

```
$ read nama
amir
$ echo $nama
```

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"
```

3. Eksekusi program `prog02.sh`

```
$ . prog02.sh
Nama Anda :
Amir
Alamat :
Jl semangka 67
Kota :
Surabaya

Hasil adalah : Amir, Jl semangka di Surabaya
```

4. Instruksi echo secara otomatis memberikan baris baru, maka untuk menghindari hal tersebut disediakan opsi –n, yang menyatakan kepada echo untuk menghilangkan baris baru. Modifikasi program `prog02.sh`

```
$ vi prog02.sh
#!/bin/sh
# prog02.sh
# membaca nama dan alamat
```

```
echo –n “Nama Anda : “
read nama
echo –n “Alamat : “
read alamat
echo –n “Kota : “
read kota

echo
echo “Hasil adalah : $nama, $alamat di $kota”
```

5. Eksekusi program `prog02.sh`

```
$ . prog02.sh
Nama Anda : Amir
Alamat : Jl semangka 67
Kota : Surabaya

Hasil adalah : Amir, Jl semangka di Surabaya
```

6. Variabel kosong adalah variable yang tidak mempunyai nilai. Variabel ini didapat atas assignment atau membaca dari keyboard atau variable yang belum didefinisikan

```
$ read nama
<CR>
$ echo $nama
$ A=
$ B=””
$ C=$A$B
$ echo $C
```

7. Variabel dapat disubtitusikan dengan hasil eksekusi dari sebuah instruksi. Pada contoh dibawah , instruksi pwd dieksekusi lebih dahulu dengan sepasang Back Quate (tanda kutip terbalik). Hasil dari eksekusi tersebut akan masuk sebagai nilai variable DIR

```
$ pwd
$ DIR=`pwd`
$ echo $DIR
```

8. Buatlah shell script `prog03.sh`

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

echo Nama Pengguna Aktif adalah $NAMA

tanggal=`date | cut –c1-10`

echo Hari ini tanggal $tanggal
```

9. Eksekusi `prog03.sh`

```
$ . prog03.sh
```

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"
```

2. Eksekusi `prog04.sh` tanpa parameter, dengan 2 parameter, dengan 4 parameter

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

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"
```

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

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

5. Membuat shell script `prog04.sh` versi 3 dengan menambahkan total parameter dan nomor proses id (PID)

```
$ 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 $$"
```

6. Eksekusi `prog04.sh` dengan 4 parameter

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

Percobaan 5 : Status Exit

1. String tidak diketemukan, maka status exit adalah 1

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

2. String diketemukan, maka status exit adalah 0

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

Percobaan 6 : Konstruksi if

1. Instruksi dengan exit status 0

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

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
```

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

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

Percobaan 7 : Konstruksi if then else

1. Membuat shell script `prog05.sh`

```
$ vi prog05.sh
#!/bin/sh
# prog05.sh
# Program akan memberikankonfirmasi 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
```

2. Jalankan `prog05.sh`, masukkan nama pemakai yang aktif yang tampil pada instruksi who dan coba juga untuk nama pemakai yang tidak aktif

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