

Pemrograman Shell



Pesantren Teknologi Informasi dan Komunikasi

Jln. Mandor Basar No. 54 RT 01/RW 01 Rangkapanjaya,
Pancoran Mas, Depok 16435 | Telp. (021) 77 88 66 91
Koordinat (-6.386680 S, 106.777305 E)

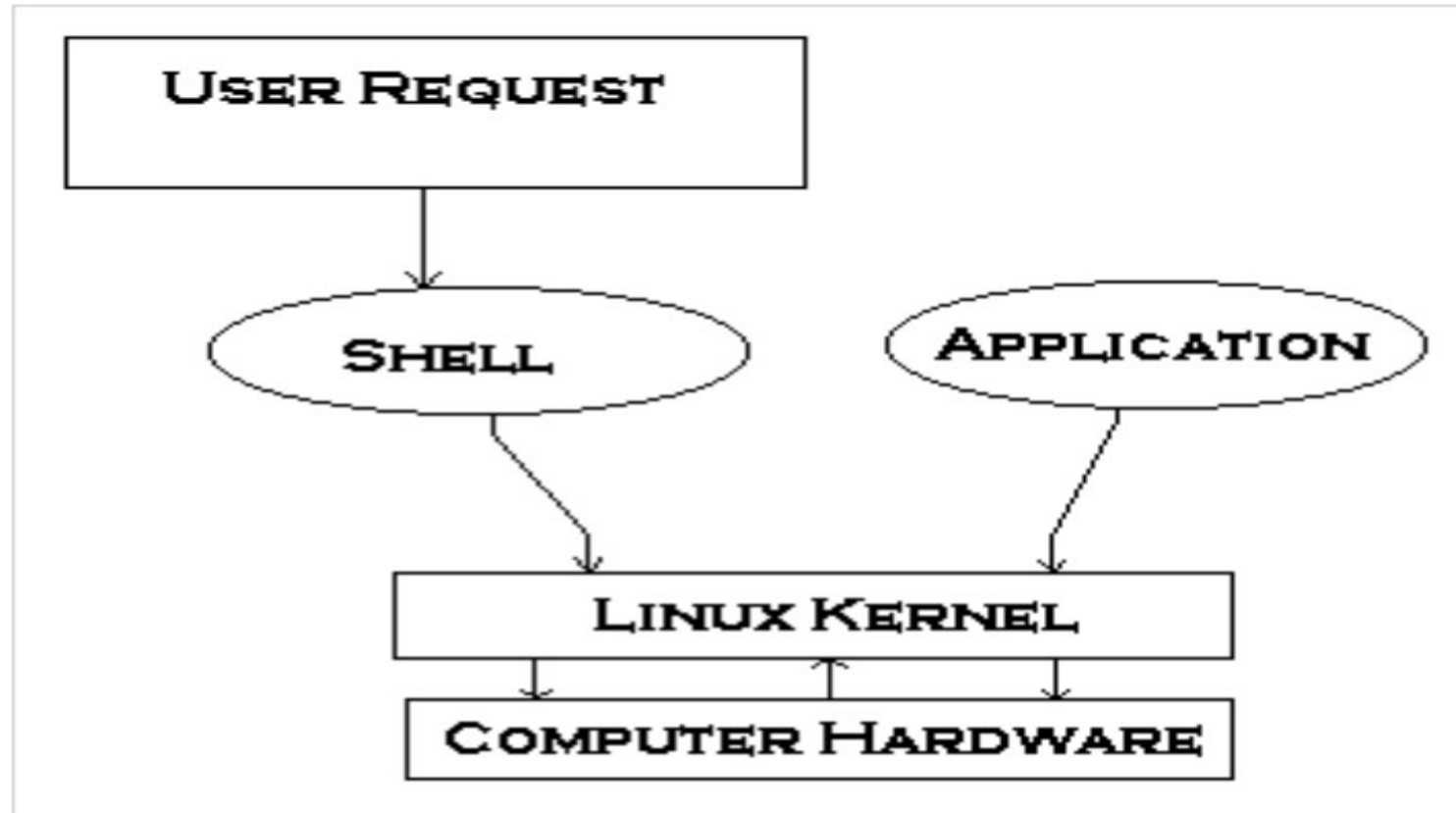
www.petik.or.id



Shell

- Antarmuka yang menjembatani antara user dengan kernel sistem operasi
- Berfungsi sebagai penterjemah perintah (command interpreter) dan bahasa pemrograman (programming language)
- Shell default pada Linux adalah bash

Posisi Shell



Shell Script

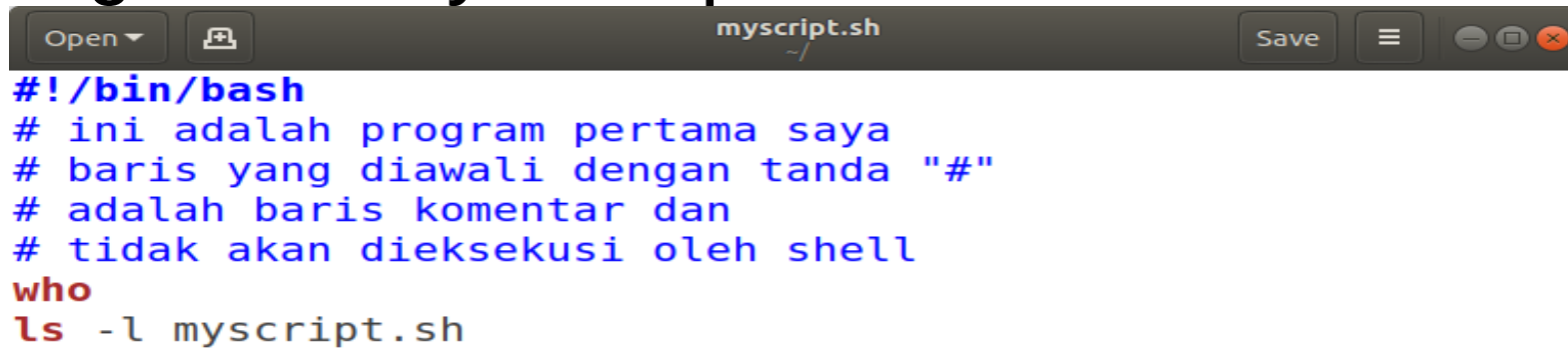
- File berisi kumpulan perintah yang harus dieksekusi oleh shell
- Jenis perintah :
 - ♦ Perintah yang diberikan dari prompt
 - ♦ Perintah kendali alir (flow control command)
- Perintah kendali alir :
 - ♦ Pencabangan
 - ♦ Pengulangan

Membuat Shell Script

- Gunakan editor teks seperti vi/vim, nano, gedit, dll
- Baris pertama diisi dengan `#!/` diikuti nama shell, contoh :
`#!/bin/bash`
- Tanda pagar (`#`) merupakan komentar dan tidak akan dieksekusi
- Nama file script umumnya diakhiri dengan `.sh`, contoh:
`myscript.sh`

Membuat Shell Script

\$ gedit myscript.sh

A screenshot of a gedit text editor window titled 'myscript.sh'. The window has a dark theme and includes standard window controls (Open, Save, menu, and window management buttons). The content of the file is as follows:

```
#!/bin/bash
# ini adalah program pertama saya
# baris yang diawali dengan tanda "#"
# adalah baris komentar dan
# tidak akan dieksekusi oleh shell
who
ls -l myscript.sh
```

Mengeksekusi Shell Script

Dieksekusi seperti executable file dengan mengetikkan nama file pada prompt

Perhatikan!

1. Path

- ♦ ditulis menggunakan absolut maupun relatif path
- ♦ masukkan direktori ke dalam search PATH

2. Permission file

- ♦ harus ada ijin akses r dan x, ubah dengan chmod jika perlu

Mengeksekusi Shell Script

```
$ myscript.sh
bash: myscript.sh: command not found
$ ./myscript.sh
bash: ./myscript.sh: Permission denied

$ echo $PATH
/usr/kerberos/bin:/usr/local/bin:/usr/bin:/bin:/usr/X11R6/bin:/home/
dudi/bin
$ mv myscript.sh bin/
$ myscript.sh
bash: /home/dudi/bin/myscript.sh: Permission denied
```


Mengeksekusi Shell Script

```
$ cd bin/
$ ls -l myscript.sh
-rw-rw-r-- 1 dudi dudi 158 Apr 21 15:52 myscript.sh
$ chmod u+x myscript.sh
$ ls -l myscript.sh
-rwxrw-r-- 1 dudi dudi 158 Apr 21 15:52 myscript.sh

$ myscript.sh
dudi      :0                2010-04-21 15:57
dudi      pts/1            2010-04-21 16:01 (:0.0)
-rwxrw-r-- 1 dudi dudi 161 Apr 21 16:14 myscript.sh
```

Variable

- Nama yang mewakili suatu harga
- Jenis variable:
 1. System variables : huruf besar
 2. User-defined variables : huruf kecil
- Nama variable harus diawali alfabet atau garis bawah
- Nama variable tidak boleh diawali oleh numerik atau karakter spesial

Variable

Contoh nama variable yang benar:

- `_no`
- `nilai_ujian`
- `nama`

Contoh nama variable yang salah:

- `5no`
- `nilai-ujian`
- `$nama`

Mendefinisikan Variable

Bentuk umum: nama_var=nilai

Keterangan:

- Sebelum dan sesudah tanda = tidak boleh ada spasi

Contoh:

```
$ nama=dudi
```

```
$ jumlah=5
```

```
$ _nilai=10
```

Menampilkan Nilai Variable

```
$ echo $nama
```

```
dudi
```

```
$ echo $jumlah
```

```
5
```

```
$ echo $_nilai
```

```
10
```

Variable Array

Mendefinisikan array:

```
$ buah=(mangga melon jeruk)
```

```
$ buah[3]=rambutan
```

```
$ buah[4]=duku
```

Menampilkan Nilai Array

```
$ echo $buah
mangga
$ echo ${buah[0]}
mangga
$ echo ${buah[2]}
jeruk
$ echo ${buah[*]}
mangga melon jeruk rambutan duku
$ echo ${buah[@]}
mangga melon jeruk rambutan duku
$ echo ${#buah[@]}
5
$ echo ${#buah[*]}
5
```

Menghapus Nilai Variable

`unset nama_var`

Contoh:

```
$ unset nama
```

```
$ unset jumlah
```

```
$ unset _nilai
```

```
$ unset buah[4]
```

```
$ unset buah
```


read

Sintaks:

```
read [-option] var1 var2 ...
```

Keterangan:

- kata pertama di-assign ke var1, kata kedua di-assign ke var2 dan seterusnya
- jika jumlah variabel lebih sedikit dibanding jumlah kata yang dibaca maka sisa baris disimpan di variabel terakhir
- option -p : tampilkan prompt tanpa baris baru

read

Contoh:

```
$ read nama
```

```
Dudi Fitriahadi
```

```
$ echo $nama
```

```
Dudi Fitriahadi
```

```
$ read -p "Nama anda : " nama
```

```
Nama anda : Unyil
```

```
$ echo $nama
```

```
Unyil
```

echo

Sintaks:

echo [-option] arg1 arg2 ...

Option:

- n : tidak menghasilkan baris baru
- e : menampilkan backslash-escaped character, yaitu:
 - \a : alert (bell)
 - \b : backspace
 - \c : suppress trailing newline
 - \n : new line
 - \r : carriage return
 - \t : horizontal tab
 - \\ : backslash

echo

Contoh:

```
$ echo
```

```
$ echo *
```

```
bin latihan latihan.txt
```

```
$ echo -e "x\ty"
```

```
x          y
```

```
$ echo -e "\n\nlompati 2 baris"
```

```
lompati 2 baris
```

echo



Contoh lain:

```
$ cat masukan1
echo "Nama anda : "
read nama
$ masukan1
Nama anda :
Unyil
$ cat masukan2
echo -e "Nama anda : \c"
read nama
$ masukan2
Nama anda : Unyil
```

Parameter Posisi

\$ perintah argumen1 argumen2 ...

\$0 \$1 \$2

Parameter Posisi

Contoh:

```
$ cat cekuser  
grep $1 /etc/passwd  
$ cekuser dudi  
dudi:x:500:500:Dudi Fitriahadi:/home/dudi:/bin/bash
```

```
$ cat ceklogin  
who | grep $1  
$ ceklogin dudi  
dudi      :0      2010-04-21 08:41  
dudi      pts/1    2010-04-21 10:13 (:0.0)  
dudi      pts/2    2010-04-21 12:04 (:0.0)
```

Parameter Spesial

\$* : berisi nilai seluruh parameter posisi
= "S1 \$2 \$3 ..."

\$@ : berisi nilai seluruh parameter posisi
= "\$1" "\$2" "\$3" ...

\$# : berisi jumlah parameter posisi

\$0 : nama program

\$? : exit status

\$\$: proses id shell

Parameter Spesial

Contoh:

```
$ cat param1
echo Nama program = $0
echo Jumlah parameter = $#
echo Parameter1=:$1: Parameter2=:$2: Parameter3=:$3:
```

```
$ param1 a b c
Nama program = /home/dudi/bin/param1
Jumlah parameter = 3
Parameter1=:a: Parameter2=:b: Parameter3=:c:
```

```
$ param1 "a b c"
Nama program = /home/dudi/bin/param1
Jumlah parameter = 1
Parameter1=:a b c: Parameter2=:: Parameter3=::
```

Parameter Spesial

Contoh:

```
$ cat param2  
echo Nama program = $0  
echo Jumlah parameter = $#  
echo Yaitu $*
```

```
$ param2 a b c  
Nama program = /home/dudi/bin/param2  
Jumlah parameter = 3  
Yaitu a b c
```

```
$ param2 "a b c"  
Nama program = /home/dudi/bin/param2  
Jumlah parameter = 1  
Yaitu a b c
```

Parameter Spesial

Contoh:

```
$ cat param3  
echo Nama program = $0  
echo Jumlah parameter = $#  
echo Yaitu $@
```

```
$ param3 a b c  
Nama program = /home/dudi/bin/param3  
Jumlah parameter = 3  
Yaitu a b c
```

```
$ param3 "a b c"  
Nama program = /home/dudi/bin/param3  
Jumlah parameter = 1  
Yaitu a b c
```

Pengutipan

Tujuan: Menghilangkan fungsi karakter spesial, seperti \$, *, ?, \

1. Backslash (\) - menghilangkan fungsi karakter spesial yang mengikuti backslash

Contoh:

```
$ echo *
ceklogin cekuser masukan1 masukan2 param1 param2
param3 tampil
$ echo \*
*
```

Pengutipan

2. Kutip Ganda (Double Quote) - menghilangkan fungsi karakter spesial yang berada dalam tanda kutip ganda kecuali \, \$ dan \

Contoh:

```
$ echo *
ceklogin cekuser masukan1 masukan2 param1 param2 param3
tampil
$ echo "*"
*

$ x=satu
$ echo "$x"
satu
```

Pengutipan

3. Kutip Tunggal (Single Quote) – menghilangkan fungsi semua karakter spesial yang berada dalam tanda kutip tunggal

Contoh:

```
$ echo '*'  
*
```

```
$ x=satu  
$ echo '$x'  
$x
```

Substitusi Perintah

Sintaks:

`$(perintah)`

atau

``perintah``

Contoh:

```
$ echo "sekarang jam $(date +%H)"
```

```
$ echo "sekarang jam `date +%H`"
```

Substitusi Nama File

Karakter	Arti
?	Match satu karakter
*	Match semua karakter
[...]	Match pilihan karakter

Substitusi Nama File

Contoh Penggunaan:

```
$ touch {T,t}es
$ touch tes{1,2}
$ ls -l
total 0
-rw-rw-r-- 1 dudi dudi 0 May  5 12:45 tes
-rw-rw-r-- 1 dudi dudi 0 May  5 12:45 Tes
-rw-rw-r-- 1 dudi dudi 0 May  5 12:46 tes1
-rw-rw-r-- 1 dudi dudi 0 May  5 12:46 tes2
$ ls tes?
tes1  tes2
$ ls tes*
tes  tes1  tes2
$ ls [Tt]es
tes  Tes
```

Seleksi dengan if

Bentuk Umum :

```
if kondisi
then
    perintah
    perintah
    :
fi
```

```
if kondisi
then
    perintah
    perintah
    :
else
    perintah
    perintah
    :
fi
```

Seleksi dengan if

Keterangan:

- Kondisi diperoleh dari hasil eksekusi program (berupa exit status)
- Exit status = 0 maka perintah-perintah antara then dan fi dilaksanakan
- Exit status \neq 0 maka:
 - ♦ Perintah-perintah antara then dan fi dilompati
 - ♦ Untuk bentuk if dengan else maka perintah-perintah antara else dan fi dilaksanakan

Seleksi dengan if

Exit Status:

- Suatu angka yang menunjukkan sukses tidaknya suatu program atau perintah dijalankan
- Exit status = 0 berarti sukses
- Exit status \neq 0 berarti gagal
- Exit status dari suatu perintah disimpan dalam variabel `$?` dan dapat ditampilkan dengan perintah:
`$ echo $?`

Seleksi dengan if

Contoh Penggunaan Exit Status:

```
$ echo $?
```

```
0
```

```
$ ls -l kosong
```

```
ls: kosong: No such file or directory
```

```
$ echo $?
```

```
2
```

Seleksi dengan if

Contoh Penggunaan if:

```
$ cat ceklogin
user="$1"
if who | grep $user > /dev/null
then
    echo "$user sedang login"
fi
```

```
$ who
dudi          :0          2010-05-05 09:27
dudi          pts/1       2010-05-05 09:47 (:0.0)
dudi          pts/2       2010-05-05 10:14 (:0.0)
$ ceklogin dudi
dudi sedang login
$ ceklogin unyil
```

Seleksi dengan if

Contoh Penggunaan if:

```
$ cat ceklogin
user="$1"
if who | grep $user > /dev/null
then
    echo "$user sedang login"
else
    echo "$user tidak login"
fi
$ ceklogin dudi
dudi sedang login
$ ceklogin unyil
unyil tidak login
```

Seleksi dengan if

Bentuk Umum if-else-if:

```
if kondisi1
then
    perintah
    perintah
    :
else
    if kondisi2
    then
        perintah
        perintah
        :
    else
        perintah
        perintah
        :
    fi
fi
```

```
if kondisi1
then
    perintah
    perintah
    :
elif kondisi2
then
    perintah
    perintah
    :
else
    perintah
    perintah
    :
fi
```


Perintah Bersyarat

1. && (logika and)

Sintaks:

```
perintah1 && perintah2
```

Keterangan:

Perintah2 dikerjakan apabila perintah1 menghasilkan exit status = 0, jika tidak maka perintah2 tidak dikerjakan

Perintah Bersyarat

2. || (logika or)

Sintaks:

```
perintah1 || perintah2
```

Keterangan:

Perintah2 dikerjakan apabila perintah1 menghasilkan exit status $\neq 0$, jika tidak maka perintah2 tidak dikerjakan

Perintah Bersyarat

Contoh Penggunaan:

```
$ cat ceklogin2
user="$1"
who | grep "$user" > /dev/null && echo "$user sedang login"
$ ceklogin2 dudi
dudi sedang login
$ ceklogin2 unyil
```

```
$ cat ceklogin3
user="$1"
who | grep "$user" > /dev/null || echo "$user tidak login"
[dudi@dudi-laptop bin]$ ceklogin3 dudi
[dudi@dudi-laptop bin]$ ceklogin3 unyil
unyil tidak login
```

Perintah Bersyarat

Contoh Penggunaan:

```
$ cat ceklogin4
user="$1"
who | grep "$user" > /dev/null && echo "$user sedang login" || echo
"$user tidak login"
$ who
dudi      :0                2010-05-05 09:27
dudi      pts/1            2010-05-05 09:47 (:0.0)
dudi      pts/2            2010-05-05 10:14 (:0.0)
$ ceklogin4 dudi
dudi sedang login
$ ceklogin4 unyil
unyil tidak login
```

Evaluasi Kondisi

Bentuk Umum:

test kondisi atau [kondisi]

Keterangan:

Untuk bentuk [kondisi] harus ada spasi setelah tanda “[” dan sebelum tanda “]”

Test mengembalikan :

Exit status = 0 jika kondisi true

Exit status \neq 0 jika kondisi false

Operator String

Operator	Keterangan
string1 = string2	string1 sama dengan string2
string1 != string2	string1 tidak sama dengan string2
-n string1	string tidak null
-z string1	string null

Operator String

Contoh Penggunaan:

```
$ nama="unyii"
$ [ "$nama" = "unyii" ]
$ echo $?
0
$ [ "$nama" != "unyii" ]
$ echo $?
1
$ [ -n "$nama" ]
$ echo $?
0
$ [ -z "$nama" ]
$ echo $?
1
```

Operator Integer

Operator	Keterangan
int1 -eq int2	int1 sama dengan int2
int1 -ne int2	int1 tidak sama dengan int2
int1 -gt int2	int1 lebih besar dari int2
int1 -ge int2	int1 lebih besar atau sama dengan int2
int1 -lt int2	int1 lebih kecil dari int2
int1 -le int2	int1 lebih kecil atau sama dengan int2

Operator Integer

Contoh Penggunaan:

```
$ x1="005"  
$ x2=" 10"  
$ [ "$x1" = 5 ]  
$ echo $?  
1  
$ [ "$x1" -eq 5 ]  
$ echo $?  
0  
$ [ "$x2" = 10 ]  
$ echo $?  
1  
$ [ "$x2" -eq 10 ]  
$ echo $?  
0
```

Operator File

Operator	Keterangan
-e file	True jika file exist
-f file	True jika file exist dan merupakan file reguler
-d file	True jika file exist dan merupakan direktori
-b file	True jika file exist dan merupakan block device
-c file	True jika file exist dan merupakan character device
-r file	True jika file exist dan readable
-w file	True jika file exist dan writeable
-x file	True jika file exist dan executable
-h atau -L	True jika file exist dan merupakan symbolic link

Operator File

Contoh Penggunaan:

```
$ ls -l
total 8
drwxrwxr-x 2 dudi dudi 4096 May 17 20:51 data
-rw-rw-r-- 1 dudi dudi 146 May 17 20:52 latihan.txt
lrwxrwxrwx 1 dudi dudi 9 May 17 20:53 uji.txt -> latihan.txt
$ [ -f latihan.txt ]
$ echo $?
0
$ [ -d data ]
$ echo $?
0
$ [ -L uji.txt ]
$ echo $?
0
```

Operator Logika

1. Operator negasi (!)

Fungsi: me-negasi-kan hasil evaluasi kondisi/ekspresi

Contoh Penggunaan:

```
$ ls -l
total 8
drwxrwxr-x 2 dudi dudi 4096 May 17 20:51 data
-rw-rw-r-- 1 dudi dudi 146 May 17 20:52 latihan.txt
lrwxrwxrwx 1 dudi dudi 9 May 17 20:53 uji.txt -> latihan.txt
$ [ ! -f latihan.txt ]
$ echo $?
1
```

Operator Logika

2. Operator and (-a)

Fungsi: melakukan fungsi logika and terhadap 2 ekspresi.
Mengembalikan hasil true jika kedua ekspresi benar

Bentuk umum:

[ekspresi1 -a ekspresi2]

Operator Logika

Contoh Penggunaan:

```
$ ls -l
total 8
drwxrwxr-x 2 dudi dudi 4096 May 17 20:51 data
-rw-rw-r-- 1 dudi dudi 146 May 17 20:52 latih.txt
lrwxrwxrwx 1 dudi dudi 9 May 17 20:53 uji.txt -> latih.txt
$ [ -f latih.txt -a -d data ]
$ echo $?
0
$ [ -f latih.txt -a -f data ]
$ echo $?
1
```

Operator Logika

3. Operator or (-o)

Fungsi: melakukan fungsi logika or terhadap 2 ekspresi.
Mengembalikan hasil true jika salah satu ekspresi benar

Bentuk umum:

`[ekspresi1 -o ekspresi2]`

Operator Logika

Contoh Penggunaan:

```
$ ls -l
total 8
drwxrwxr-x 2 dudi dudi 4096 May 17 20:51 data
-rw-rw-r-- 1 dudi dudi  146 May 17 20:52 latih.txt
lrwxrwxrwx 1 dudi dudi    9 May 17 20:53 uji.txt -> latih.txt
$ [ -f latih.txt -o -f data ]
$ echo $?
0
$ [ -f latih.txt -o -d data ]
$ echo $?
0
$ [ -d latih.txt -o -f data ]
$ echo $?
1
```


Ekspresi Aritmatika

Operator	Keterangan
+	penjumlahan
-	pengurangan
/	pembagian
*	perkalian
%	Modulo (siswa pembagian)

Keterangan:

- "+" dan "-" memiliki prioritas lebih rendah dibanding tiga operator lain
- Setiap operator didahului dan diikuti oleh spasi
- Operator "*" harus dikutip agar tidak rancu dengan substitusi file

Ekspresi Aritmatika

- `expr`

Bentuk umum: `expr` ekspresi

Contoh:

```
$ expr 3 + 2
```

```
5
```

```
$ expr 5 - 2
```

```
3
```

```
$ expr 5 \* 2
```

```
10
```

```
$ expr 5 / 2
```

```
2
```

```
$ expr 5 % 2
```

```
1
```

Ekspresi Aritmatika

Bila hasil perintah `expr` akan disimpan ke dalam suatu variabel maka perintah `expr` harus berada dalam substitusi perintah

Contoh:

```
$ i=1
$ i=`expr $i + 1`
$ echo $i
2
```

```
$ i=1
$ i=$(expr $i + 1)
$ echo $i
2
```

Ekspresi Aritmatika

- let

Bentuk umum: let ekspresi

Contoh:

```
$ i=1  
$ let i=$i+1  
$ echo $i  
2
```

Ekspresi Aritmatika

- (())

Bentuk umum: ((ekspresi))

Contoh:

```
$ i=1  
$ ((i=$i+1))  
$ echo $i  
2
```

Perulangan dengan for



Bentuk Umum:

```
1. for dengan list
   for var in word
   do
       perintah
   perintah
   :
done
```

Perulangan dengan for



```
2. for tanpa list
    for var
    do
        perintah
    perintah
    :
done
```

Perulangan dengan for

3. Bentuk for yang lain
- ```
for ((expr1; expr2; expr3;))
do
 perintah
perintah
:
done
```
- Keterangan:  
expr1 = nilai awal  
expr2 = kondisi yg dievaluasi  
expr3 = increment



# Perulangan dengan for



## Contoh for dengan list:

```
$ cat loop1
for i in 1 2 3
do
 echo $i
done
$ loop1
1
2
3
```

# Perulangan dengan for

## Contoh for tanpa list:

```
$ cat loop2
for i
do
 echo $i
done
$ loop2 1 2 3
1
2
3
```

# Perulangan dengan for

Contoh for yang lain:

```
$ cat loop3
for ((i=1; i<=3; i++))
do
 echo "$i"
done
$ loop3
1
2
3
```

# Seleksi dengan case

Bentuk Umum:

case nilai in

pola\_1 ) perintah ;;

pola\_2 ) perintah ;;

pola\_3 ) perintah ;;

:

esac

# Seleksi dengan case

- case berfungsi untuk membandingkan satu nilai dengan sejumlah pola
- case membandingkan nilai dengan pola yang ada dari atas ke bawah
- Jika ada pola yang terpenuhi maka perintah untuk pola tersebut dieksekusi
- Jika tidak ada pola yang terpenuhi maka tidak ada perintah dalam case yang dieksekusi
- case dapat menggunakan sejumlah pola karakter seperti: ?, [ ...] dan \*

# Seleksi dengan case

## Contoh:

```
$ cat hari_ini
kode_hari=`date +%w`
case "$kode_hari" in
 0) echo "Hari ini hari Minggu" ;;
 1) echo "Hari ini hari Senin" ;;
 2) echo "Hari ini hari Selasa" ;;
 3) echo "Hari ini hari Rabu" ;;
 4) echo "Hari ini hari Kamis" ;;
 5) echo "Hari ini hari Jumat" ;;
 6) echo "Hari ini hari Sabtu" ;;
esac
$ date
Mon May 30 19:29:20 WIT 2011
$ hari_ini
Hari ini hari Senin
```

# Seleksi dengan case

## Contoh:

```
$ cat ctype
LC_COLLATE=C
read -p "Ketik satu karakter : " char
case "$char" in
 [0-9]) echo "$char : angka" ;;
 [a-z]) echo "$char : huruf kecil" ;;
 [A-Z]) echo "$char : huruf besar" ;;
 ?) echo "$char : karakter khusus" ;;
 *) echo "$char : ketikkan 1 karakter saja" ;;
esac
unset LC_COLLATE
```

# Seleksi dengan case

## Contoh Eksekusi:

```
$ ctype
```

```
Ketik satu karakter : a
```

```
a : huruf kecil
```

```
$ ctype
```

```
Ketik satu karakter : G
```

```
G : huruf besar
```

```
$ ctype
```

```
Ketik satu karakter : >
```

```
> : karakter khusus
```

```
$ ctype
```

```
Ketik satu karakter : df
```

```
df : ketikkan 1 karakter saja
```



# Seleksi dengan case

## Contoh:

```
$ cat salam
hour=`date +%H`
case "$hour" in
 0? | 1[01]) echo "Good morning" ;;
 1[2-7]) echo "Good afternoon" ;;
 *) echo "Good evening" ;;
esac
$ date
Mon Jun 14 14:07:46 WIT 2010
$ salam
Good afternoon
```

# Perulangan dengan while



Bentuk Umum:

while kondisi

do

    perintah

    perintah

  :

done

# Perulangan dengan while

## Contoh:

```
$ cat ulang1
i=1
while [$i -le 5]
do
 echo $i
 i=`expr $i + 1`
done
$ ulang1
1
2
3
4
5
```

# Perulangan dengan while

- true

Perintah true digunakan dalam while apabila menginginkan pengulangan secara terus menerus

Contoh:

```
$ cat ulang2
while true
do
 echo "Hallo; hit [CTRL+C] to stop!"
done
$ ulang2
Hallo; hit [CTRL+C] to stop!
Hallo; hit [CTRL+C] to stop!
Hallo; hit [CTRL+C] to stop!
```

# Perulangan dengan until



Bentuk Umum:

until kondisi

do

    perintah

    perintah

    :

done

# Perulangan dengan until

## Contoh:

```
$ cat ulang3
i=1
until [$i -gt 5]
do
 echo $i
 i=`expr $i + 1`
done
$ ulang3
1
2
3
4
5
```

# Perulangan dengan until

- false

Perintah false digunakan dalam until apabila menginginkan pengulangan secara terus menerus

Contoh:

```
$ cat ulang4
until false
do
 echo "Hallo; hit [CTRL+C] to stop!"
done
$ ulang4
Hallo; hit [CTRL+C] to stop!
Hallo; hit [CTRL+C] to stop!
Hallo; hit [CTRL+C] to stop!
```

# break dan continue

- break

Perintah break digunakan untuk keluar dari proses suatu loop

- continue

Perintah continue digunakan untuk melompati sisa perintah dalam loop untuk kembali melanjutkan loop dari atas



# break dan continue

## Contoh break:

```
$ cat potong
i=0
while [$i -le 5]
do
 i=`expr $i + 1`
 if [$i -eq 3]
 then
 break
 fi
 echo $i
done
$ potong
1
2
```

# break dan continue

## Contoh continue:

```
$ cat lompati
i=0
while [$i -le 5]
do
 i=`expr $i + 1`
 if [$i -eq 3]
 then
 continue
 fi
 echo $i
done
$ lompati
1
2
4
5
6
```

# I/O Redirection dlm pengulangan

## Contoh:

```
$ cat ulang5
i=1
while [$i -le 5]
do
 echo $i
 i=`expr $i + 1`
done > hasil.txt
$ ulang5
$ cat hasil.txt
1
2
3
4
5
```

# I/O Redirection dlm pengulangan

Contoh:

Mengambil input dari file dan menampilkannya ke layar

```
$ cat data.txt
dudi 89
dedi 78
didi 67
$ cat ulang6
while read nama nilai
do
 echo -e "$nama\t$nilai"
done < data.txt
$ ulang6
dudi 89
dedi 78
didi 67
```

# select

Fungsi: untuk membuat menu sederhana

Sintaks:

```
select var [in list]
```

```
do
```

```
 blok-perintah
```

```
done
```

# select

- Setiap item dalam list akan ditampilkan berupa menu disertai nomor
- Nomor yang dipilih akan dimasukkan ke variabel `REPLY`, sedangkan nilai `var` akan diisi dengan nilai item dalam list
- Jika pilihan valid maka blok-perintah akan dieksekusi
- `Select` akan mengulang pilihan sampai ada perintah yang menghentikannya misalnya: `break`
- Prompt default adalah `#?`, apabila akan diganti maka harus mengubah nilai variabel `PS3`

# select

## Contoh:

```
$ cat menu1
```

```
select pilih in tambah lihat hapus;
```

```
do
```

```
 echo "Anda memilih $pilih" \($REPLY\)
```

```
 break;
```

```
done
```

```
$ menu1
```

```
1) tambah
```

```
2) lihat
```

```
3) hapus
```

```
#? 1
```

```
Anda memilih tambah (1)
```

# select



## Contoh:

```
$ cat menu2
select pilih
do
 echo "Anda memilih $pilih" \($REPLY\)
 break;
done
$ menu1 tambah lihat hapus
1) tambah
2) lihat
3) hapus
#? 1
Anda memilih tambah (1)
```



# select

## Contoh:

```
$ cat menu3
PS3="Pilihan anda : "
echo "Menu yang tersedia : "
echo "-----"
select pilih in tambah lihat hapus;
do
 echo "Anda memilih $pilih" \($REPLY\)
 break;
done
$ menu3
Menu yang tersedia :

1) tambah
2) lihat
3) hapus
Pilihan anda : 2
Anda memilih lihat (2)
```

# Fungsi

Sintaks:

```
function nama_fungsi {
 blok-perintah
}
```

atau

```
nama_fungsi() {
 blok-perintah
}
```

# Fungsi

Contoh:

```
$ cat fungsi
nu () { who | wc -l; }
$. fungsi
$ who
dudi :0 2010-07-07 17:33
dudi pts/1 2010-07-07 17:34 (:0.0)
$ nu
2
```



Jalan Mandor Basar Nomor 54, RT. 01/001, Rangkapanjaya, Pancoran  
Mas, Kota Depok 16435



[www.petik.or.id](http://www.petik.or.id)



021 7788 6691



[info@petik.or.id](mailto:info@petik.or.id)