Praktikum Sistem Operasi

(Pertemuan 06)

Pemrograman Shell

Husni

husni@mail.ugm.ac.id

Garis Besar

- Mengenal Shell Scripting
- Seleksi & Perulangan
- Parameter dan Opsi
- Scripting Lanjut

Mengenal Shell Scripting

- Script Pertama: haloo
- Komentar
- Variabel
- Agar Variabel dikenal di Luar Script
- Troubleshooting a script
- Latihan

Script Pertama: haloo

- **Catatan**: pastikan anda telah menguasai dengan baik materi tentang Maksimalisasi Shell dan Pipe & Filtering
- Contoh: file script bernama haloo

```
echo echo Saya sedang belajar Linux > haloo
cat haloo
echo Saya sedang belajar Linux
chmod +x haloo   ← menjadikan executable
./haloo          ← mengeksekusi script
Saya sedang belajar Linux
```

She-Bang

- String **#!/bin/bash** sering diletakkan pada baris pertama setiap script Shell
- **#!** - dibaca she-bang adalah 2 karakter yang mengawali script shell
- Maksudnya, baris-baris script di bawah #!/bin/bash akan dieksekusi menggunakan program shell bash, bukan shell lain
- cat > script02
- #!/bin/bash
  echo Dua baris terakhir dari /etc/passwd:
  echo `tail -2 /etc/passwd`
- ./script02
  Dua baris terakhir dari /etc/passwd:
  mysql:x:115:126:MySQL Server,,,:/nonexistent:/bin/false
  jetty:x:116:127::/usr/share/jetty:/bin/false

Komentar

- Setiap karakter setelah tanda pound # dianggap komentar
- Selain baris pertama, jika setelah she # diikuti bang !
- Contoh

```
#!/bin/bash
# script: script03
# Hanya menampilkan nama user dan hostname
#
echo $USER login pada $HOSTNAME
```

- chmod +x script03
- ./script03

  husni login pada husni-Z475

Variabel

- Pembuatan variabel pada script shell sama seperti pembuatan variabel langsung pada shell
- Contoh

```
#!/bin/bash

# script04

# menggunakan variabel

var1=”Joko Susanto”

echo Nilai var1 = $var1
```

- Variabel di dalam script tidak dikenali di luar (di Shell Linux)

```
echo $var1
```

Agar Variabel dikenal di Luar Script

- Dikenal dengan istilah *Sourcing The Script*
- Variabel di dalam script, dapat dikenali langsung dari shell
- Contoh

```
./script04
Nilai var1 = Joko Susanto
echo $var1

source ./script04
Nilai var1 = Joko Susanto
echo $var1
Joko Susanto
```

Cara Lain:

```
. ./script04
Nilai var1 = Joko Susanto

echo $var1
Joko Susanto
```

Troubleshoot Terhadap Script

- File script juga dapat dieksekusi dengan memanggil program bash

  bash script03

  husni login pada husni-Z475

- Tahapan eksekusi, *step by step*, dapat diketahui memanfaatkan opsi -x. Ini bermanfaat men-debug script atau mencari posisi kesalahan dalam script

  bash -x script03

  + echo husni login pada husni-Z475

  husni login pada husni-Z475

Latihan

- Buat sebuah script yang mendefinisikan dua variabel dan menampilkan outputnya
- Buatlah agar script sebelumnya berpengaruh terhadap shell yang sedang digunakan
- Adakah cara lebih singkat untuk meng-*source the script*?
- Berikan komentar di dalam script tersebut sehingga lebih mudah dipahami oleh pemrogram pemula sekalipun
- Buat sebuah script yang mampu menampilkan hostname, IP address, subnet mask dan gateway dari komputer yang anda gunakan!

Seleksi & Perulangan

- Test[ ]
- If then else
- If then elif
- Perulangan for
- Perulangan while
- Perulangan until
- Latihan

Test[ ]

- Perintah test digunakan untuk menguji apakah sesuatu bernilai true atau false.
- Apakah 17 lebih besar daripada 45?

  test 17 -gt 45 ; echo $?

  1

  Hasilnya adalah 1, berarti false (salah)
- Apakah 17 lebih kecil daripada 45?

  test 17 -lt 45 ; echo $?

  0

  Nilai 0 menunjukkan true (benar)

Test[ ]

- Jika nilai numerik (0 dan 1) tidak nyaman, kita dapat memodifikasi bentuk perintah sehingga output berupa string “benar” atau “salah”

  test 17 -lt 45 && echo Benar || echo Salah

  Benar

  test 17 -gt 45 && echo Benar || echo Salah

  Salah

- Pahami kembali peran operator dalam operasi shell Linux

Contoh test[] - Lihat man test

- [-d foo] Adakah direktori bernama foo?
- [-e bar] Adalah file bernama bar?
- ['/etc' = $PWD] Apakah string /etc sama dengan variabel $PWD?
- [$1 != 'rahasia'] Apakah parameter pertama berbeda dengan rahasia?
- [55 -lt $bar] Apakah 55 kurang dari nilai $bar?
- [$foo -ge 1000] Apakah nilai $foo lebih atau sama dengan 1000
- ["abc" < $bar] Apakah urutan abc sebelum nilai dari $bar
- [-f foo] Apakah foo suatu file biasa?
- [-r bar] Apakah bar suatu file readable
- [foo -nt bar] Apakah file foo lebih baru daripada file bar?
- [-o nounset] Apakah opsi Shell nounset diset?

Pemanfaatan Kombinasi AND & OR

- [-d Public] && echo Ada || echo Tidak ada

  Ada
- ['/etc' = $PWD] && echo Iya || echo Bukan

  Bukan
- [66 -gt 55 -a 66 -lt 500] && echo true || echo false

  true
- [66 -gt 55 -a 660 -lt 500] && echo true || echo false

  false
- [66 -gt 55 -o 660 -lt 500] && echo true || echo false

  true

if then else

- Bagaimana menentukan pilihan
- Jika kondisi tertentu terpenuhi maka lakukan sesuatu, jika tidak maka lakukan yang lain
- Contoh: pilihan01

  Tampilkan pesan ada tidaknya file test5.doc di direktori aktif?

  ```
  #!/bin/bash
  if [ -f test5.doc ]
  then echo File test5.doc ADA!
  else echo File test5.doc TIDAK ADA!
  fi
  ```

- chmod +x pilihan01
- ./pilihan01

  File test5.doc ADA!

if then elif

- If bersarang di dalam else
- Contoh: pilihan02

```
#!/bin/bash
# file script: pilihan02
nilai=99
if [ $nilai -eq 23 ]
then
echo "23 TEPAT dan PAS."
elif [ $nilai -gt 23 ]
then
echo "TERLALU BANYAK."
else
echo "TIDAK CUKUP."
fi
```

- chmod +x pilihan02
- ./pilihan02

  TERLALU BANYAK.

Perulangan for

- Lakukan perulangan SEBANYAK yang ditentukan

```
#!/bin/bash
# file script: for01
for i in 1 2 4
do
  echo $i
done
```

- chmod +x for01
- ./for01

```
1
2
4
```

Perulangan for

- Contoh: Bentuk lain dari for (*range*)

```
#!/bin/bash
# file script: for02
for counter in {1..20}
do
  echo Menghitung 1 sampai 20, sekarang: $counter
sleep 1
done
```

- Eksekusi: (apa hasilnya?)

```
chmod +x for02
./for02
```

Perulangan while

- Lakukan perulangan SELAMA kondisi terpenuhi

```
#!/bin/bash
# file script: while01
i=100;
while [ $i -ge 0 ] ;
do
  echo Menghitung mundur, dari 100 s.d 0, Sekarang $i;
let i--;
done
```

- Bagaimana cara mengeksekusi script ini?

Perulangan until

- Lakukan perulangan SAMPAI kondisi terpenuhi

```
#!/bin/bash
# file script: until01
let i=100;
until [ $i -le 0 ] ;
do
  echo Menghitung mundur, dari 100 s.d 1, Sekarang $i;
let i--;
done
```

- Coba eksekusi script until01 di atas!

Latihan

- Tuliskan script yang menggunakan for untuk menghitung dari 1 s.d 17500
- Tuliskan script yang mengguakan while untuk menghitung 3 s.d 9
- Tuliskan script yang menggunakan until untuk menghitung mundur dari 8 s.d 3
- Tuliskan script yang menghitung jumlah dari file yang berakhiran .txt dalam direktori aktif anda
- Bagaimana jika tidak ditemukan file berakhiran .txt? BETULKAN!

Parameter dan Opsi

- Parameter Script
- Shift
- Input runtime: read
- Men-source-kan file config
- getopts
- Latihan

Parameter Script

- Script shell bash dapat mempunyai parameter
- Contoh

```
#!/bin/bash
# file script: par01
echo 'Parameter pertama:' $1
echo 'Parameter kedua   :' $2
echo 'Parameter ketiga   :' $3
echo \$ $$ - PID dari script ini
echo \# $# - Jumlah parameter
echo \? $? - Kode kembalian terakhir
echo \* $* - Semua parameter
```

- chmod +x par01
- ./par01 Satu Dua Tiga

```
Parameter pertama: Satu
Parameter kedua   : Dua
Parameter ketiga   : Tiga
$ 15530 - PID dari script ini
# 3 - Jumlah parameter
? 0 - Kode kembalian terakhir
* Satu Dua Tiga - Semua parameter
```

Parameter Script

- Parameter atau argumen pertama? $1
- Nama file script? $0
- Contoh

  cat > par02

  echo File script ini bernama $0

  ./par02

  bash: ./par02: Permission denied

  chmod +x par02

  ./par02

  File script ini bernama ./par02

Pernyataan shift

- Digunakan untuk memparse semua parameter satu demi satu
- Contoh

```
#!/bin/bash
# file script: par03
if [ "$#" == "0" ]
then
  echo Sertakan minimal satu parameter.
  exit 1
fi
while (( $# ))
do
  echo Anda telah memberikan $1
  shift
done
```

- chmod +x par03
- ./par03
  Sertakan minimal satu parameter.
- ./par03 Satu Demi Kamu
  Anda telah memberikan Satu
  Anda telah memberikan Demi
  Anda telah memberikan Kamu
- ./par03 "S2 Ilmu Komputer UGM"
  Anda telah memberikan S2 Ilmu Komputer UGM

Input Runtime

- Saat script berjalan, input dari pengguna dapat diminta dengan pernyataan read
- Contoh

```
#!/bin/bash
# file script: run01
echo -n Masukkan suatu angka:
read angka
echo Angka yang dimasukan adalah $angka
```

- chmod +x run01
- ./run01

```
Masukkan suatu angka:24
Angka yang dimasukan adalah 24
```

Input Runtime

- Contoh

  ```
  #!/bin/bash
  # file script: run02
  echo 'Masukkan nama awal'
  echo -n 'diikuti nama akhir:'
  read namaawal namaakhir
  echo "Halooo $namaawal $namaakhir"
  ```

- chmod +x run02
- ./run02

  Masukkan nama awal

  diikuti nama akhir:Susi Susanti

  Halooo Susi Susanti

Menjadikan file config sebagai source

- Sering digunakan untuk men-*source*-kan file konfigurasi
- Contoh: Menggunakan isi file myApp.conf di dalam file script shell myApp.sh

```
# File konfigurasi dari script myApp

# nama file: myApp.conf

# Tentukan path di sini

myAppPath=/var/myApp

# Tentukan jumlah pengguna

users=5
```

Menjadikan file config sebagai source

- Contoh pemanggilan

```
#!/bin/bash
# file script: myApp.sh
# Welcome to the myApp application
. ./myApp.conf
echo Ada sebanyak $users pengguna
```

- chmod +x myApp.sh
- ./myApp.sh

  Ada sebanyak 5 pengguna

getopts

- Digunakan untuk mendapatkan opsi-opsi yang disertakan saat memanggil shell script
- Contoh

```
#!/bin/bash
# file script: get01
while getopts ":afz" option;
do
  case $option in
   a)
     echo received -a
   ;;
   f)
     echo received -f
   ;;
   z)
     echo received -z
   ;;
   *)
     echo "invalid option -$OPTARG"
   ;;
  esac
done
```

- chmod +x get01
- ./get01 -afZX

```
received -a
received -f
invalid option -Z
invalid option -X
```

getopts

- Suatu opsi juga dapat diharusnya menyertakan parameter
- Contoh Eksekusi
- chmod +x get02
- ./get02 -a -f Husni

  diterima -a

  diterima -f dengan Husni
- ./get02 -a -f Husni -z

  diterima -a

  diterima -f dengan Husni

  diterima -z
- ./get02 -a -f

  diterima -a

  Opsi -f memerlukan parameter.

getopts

```
#!/bin/bash
# file script: get02
while getopts ":af:z" opsi;
do
  case $opsi in
  a)
    echo diterima -a
  ;;
  f)
    echo diterima -f dengan $OPTARG
  ;;
  z)
    echo diterima -z
  ;;
  :)
    echo "Opsi -$OPTARG memerlukan parameter."
  ;;
  *)
    echo "Opsi tidak valid -$OPTARG"
  ;;
  esac
done
```

Latihan

- Tulis suatu script yang menerima dua parameter (berupa nama file) dan output-kan ada atau tidaknya kedua file tersebut
- Tulis suatu script yang meminta masukan nama file. Pastikan ada tidaknya file tersebut. Apakah file tersebut *writable*? Jika tidak buat agar *writeable*.
- Buat suatu script myApp2.sh yang melibatkan file konfigurasi myApp2.conf. Pastikan script meminta input dari pengguna. Simpan setiap interaksi pengguna ke dalam file myApp2.log

Scripting Lanjut

- Apa kegunaan empat pernyataan berikut?

  eval

  (( ))

  let

  case

- Bagaimana membuat fungsi shell?
- Kerjakan soal-soal pada halaman Latihan

Latihan

- Tulis suatu *script* yang meminta input dua bilangan dan output-kan hasil penjumlahan dan perkaliannya
- Perbaiki script no.1 untuk memastikan bahwa bilangan yang dimasukkan hanya 1 s.d 100, jika tidak maka keluar dengan pesan error.
- Perbaiki script no.1. Beritahukan pengguna jika hasil penjumlahan sama dengan hasil perkalian
- Tulis suatu script dengan melibatkan pernyataan case yang insensitive, gunakan opsi shopt nocasematch. Opsi nocasematch berguna me-reset ke nilai yang dipegang sebelum script dijalankan.
- Pelajari script sistem Linux dalam /etc/init.d dan /etc/rc.d dan coba pahami.