



LABORATORIUM JARINGAN KOMPUTER
FAKULTAS ILMU KOMPUTER
UNIVERSITAS BRAWIJAYA

PRAKTIKUM SISTEM OPERASI

SEMESTER : GENAP

TAHUN : 2015/2016

BAB V

JUDUL BAB	: USING SHELL
DISUSUN OLEH	: MOH ARIF ANDRIAN
NIM	: 156150600111002
ASISTEN	: SISKAPERMATASARI ZAENAL KURNIAWAN
KOORDINATOR ASISTEN	: DANY RAHMANA



LAPORAN PRAKTIKUM SISTEM OPERASI

FAKULTAS ILMU KOMPUTER

UNIVERSITAS BRAWIJAYA

Nama : Moh. Arif Andrian
NIM : 156150600111002
Laporan : BAB V
Asisten : Siska Permatasari
Zaenal Kurniawan

BAB V

USING SHELL

1. Dasar Teori

1.1. Shell

Shell merupakan *command executive*, yaitu program yang menunggu instruksi atau perintah dari pemakai, memeriksa sintaks dari perintah yang diberikan, dan kemudian mengeksekusi perintah tersebut.

Shell disebut juga dengan *command interpreter language*, artinya shell adalah bahasa yang menerjemahkan perintah yang menjadi penghubung secara langsung atau antar muka antara user dengan utilitas & bahasa pemrograman dari sistem operasi itu sendiri. Contohnya pada saat kita mengetikkan perintah “ls” di terminal, shell akan menjalankan perintah tersebut dengan mengeksekusi program “ls” yang ada pada sistem operasi. Jika kita mengetikkan perintah yang tidak memiliki program didalam sistem operasi, misal kita mengetikkan “hello” saja pada terminal, maka shell tidak akan mengeksekusi program apapun, dan biasanya akan muncul pesan “command not found”.

#bagaimana cara menampilkan shell yang tersedia di linux anda? tunjukkan screenshot!

Buka terminal dengan menekan ctrl+alt+t.
Ketik
\$ cat /etc/shells

1.2 Macam-macam Shell

Shell sendiri terdiri dari berbagai macam dengan kelebihan dan kekurangan

```
andrian@156150600111002: ~  
andrian@156150600111002:~$ cat /etc/shells  
# /etc/shells: valid login shells  
/bin/sh  
/bin/dash  
/bin/bash  
/bin/rbash  
andrian@156150600111002:~$
```

masing-masing. Untuk melihat langsung shell apa saja yang ada di linux bisa

dengan menggunakan perintah `$ls /etc/shells` kemudian enter. Berikut adalah beberapa contoh macam shell yang digunakan pada linux :

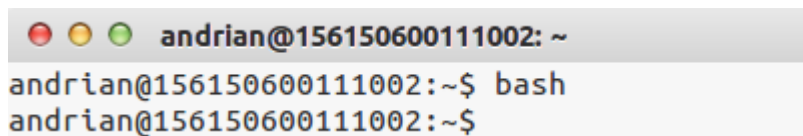
Shell	Developer	Lokasi	Keterangan
Bourne Shell (/bin/sh)	Steven Bourne		Shell yang pertama kali ada di Unix
Bourne Again Shell (/bin/bash)	Brian Fox dan Chet Ramey	Free Software Foundation	Shell BASH merupakan shell yang paling umum digunakan di semua system operasi Linux
C Shell (/bin/csh)	Bill Joy	Universitas Berkeley California	C Shell adlah Shell yang mengadopsi perintah pemrograman C
Tenec C Shell (/bin/tcsh)			TCSH adalah Shell pengembangan dari UNIX C Shell
Korn Shell (/bin/ksh)	Bill Joy	AT&T Labs	Shell ini menggabungkan kelebihan Bourne shell dan C shell sehingga lebih interaktif.
Alquimist Shell (/bin/ash)	Kenneth Almquist		Ash banyak digunakan dalam embedded linux, sistem yang tertanam misalnya BIOS.
Z Shell (/bin/zsh)			Z shell ini biasa digunakan oleh pengguna linux untuk membangun script yang kompleks.

**#ketikkan perintah "bash" pada terminal anda, apa yang akan muncul?
mengapa muncul seperti itu?**

Tidak terjadi apa-apa. Hanya saja akan berganti shell ke bash. Karena perintah “bash” berfungsi mengganti shell yang aktif sebelumnya menjadi shell bash

1.3 Shell Programming

Apa itu shell programming? Di dalam shell programming kita akan



```

andrian@156150600111002: ~
andrian@156150600111002:~$ bash
andrian@156150600111002:~$

```

menyusun atau mengelompokkan beberapa perintah shell internal maupun

eksternal sehingga menjadi kumpulan perintah yang melakukan tugas-tugas tertentu sesuai dengan apa yang dikehendaki oleh programmer atau user. Dengan kata lain shell dapat digunakan untuk membuat sebuah program layaknya bahasa pemrograman. Shell programming dapat digunakan dalam dua modus, yaitu:

- Modus interaktif

Modus interaktif adalah cara yang digunakan untuk menginputkan perintah secara langsung pada Command Line Interface perbaris. Shell akan menjalankan baris perintah yang kita inputkan setelah menekan enter.

- Modus script

Modus script adalah cara yang digunakan untuk menuliskan script atau kode program menggunakan text editor yang tersedia pada linux kemudian menyimpan file tersebut dengan ekstensi .sh. File .sh tersebut bisa kita jalankan melalui Command Line pada linux.

I.1. Mengganti dan Menjalankan Shell

Linux menggunakan bash sebagai shell default, tetapi pengguna bisa mengubah shell default untuk tiap user-nya. Untuk melihat shell yang sedang digunakan oleh user bisa dilihat pada file :

```
/etc/passwd
.....
root:x:0:0:root:/root:/bin/bash
daemon:x:1:1:daemon:/usr/sbin:/bin/sh
.....
```

Isi dari file tersebut tiap barisnya dibagi menjadi tujuh bagian, dan setiap terakhir digunakan untuk mendefinisikan shell yang digunakan. Dalam contoh di atas user root menggunakan shell Bash sedangkan daemon menggunakan shell bourne shell.

Cara lain yang bisa digunakan untuk melihat shell adalah dengan melihat environmet user dengan menjalankan perintah env. Environment user merupakan lingkungan user yang berisi semua variabel atau ketentuan khusus untuk user tersebut.

```
$ env
.....
SHELL=/bin/bash
.....
```

Ada beberapa cara yang dapat digunakan untuk mengubah shell default, yaitu dengan utilitas userconf, atau dapat juga dengan mengubah pada file /etc/passwd tersebut tetapi harus sebagai root.

#Contoh mengganti shell dengan utilitas chsh (change shell) :

- I. Ketik `$ chsh`
- II. Masukkan password username
- III. Ketikkan login shell yang diinginkan.

```
andrian@156150600111002: ~  
andrian@156150600111002:~$ chsh  
Password:  
Changing the login shell for andrian  
Enter the new value, or press ENTER for the default  
Login Shell [/bin/bash]: /bin/sh  
andrian@156150600111002:~$
```

I.1.2. Menjalankan Shell

sebuah shell dapat dijalankan tanpa harus mengubah default shell-nya. Cara yang digunakan adalah dengan memanggil nama shell pada command prompt. Misalnya, user akan menggunakan shell sh, user tinggal menjalankan shell sh, user tinggal menjalankan sh sehingga prompt akan berubah sesuai dengan prompt Bourne shell. Untuk keluar dan kembali ke shell default, ketik exit atau tekan **Ctrl + d**.

Contoh :

```
bash@praktikum:~$ sh  
$
```

```
andrian@156150600111002: ~  
andrian@156150600111002:~$ sh  
$  
andrian@156150600111002:~$ _
```

I.2. Menjalankan Script Shell

Untuk menjalankan sebuah script shell, Ada dua cara yang digunakan untuk menjalankan sebuah shell script, yaitu

- a) Langsung running dari file [dot]sh-nya.

```
$ bash hello.sh
```

```
andrian@156150600111002: ~  
andrian@156150600111002:~$ bash hello.sh  
  
Masukkan Nama anda : Andrian  
  
Hello Andrian  
  
andrian@156150600111002:~$ █
```

b) Ubah Hak Aksesnya.

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

```
andrian@156150600111002: ~  
andrian@156150600111002:~$ ls -l hello.sh  
-rw-rw-r-- 1 andrian andrian 82 Apr 19 10:00 hello.sh  
andrian@156150600111002:~$ chmod +x hello.sh  
andrian@156150600111002:~$ ls -l hello.sh  
-rwxrwxr-x 1 andrian andrian 82 Apr 19 10:00 hello.sh  
andrian@156150600111002:~$
```

1.4 Variabel

Variabel adalah tempat atau nilai yang berfungsi untuk menyimpan sebuah data. Shell memungkinkan user untuk membuat, mendefinisikan dan menghapus variabel.

1.4.1 Macam-macam Variabel

Variabel Lokal

Variabel lokal adalah variabel yang nilainya langsung diisikan. Variabel yang ada hanya pada saat masih aktif, dan hanya dikenal di lingkungan itu sendiri, sehingga variabel lokal hanya berlaku pada lingkungan dimana variabel tersebut dibuat. Tipe nilai dari variabel ini bisa bermacam-macam, ada yang bertipe numerik dan ada juga yang bertipe karakter atau alphanumerik.

Di dalam memberikan nama sebuah variabel, yang harus diperhatikan adalah nama variabel berupa karakter alphanumeric atau bisa juga diawali dengan karakter garisbawah(_), setelah itu baru bisa diikuti dengan jenis karakter yang lain.

Contoh penamaan variabel :

Penamaan variabel yang salah	Penamaan variabel yang benar
5nama	Nama
4_hasil	_hasil4
\nilai	_nilai

Dalam pemrograman Shell tidak perlu memberikan jenis variabel karena hanya ada satu jenis yaitu tipe string. Contoh dalam penulisan variabel dan pemberian nilai dalam potongan program Shell sebagai berikut :

```
#!/bin/bash
HELLO=Hello
function hello {
    local HELLO=World
    echo $HELLO
}
echo $HELLO
hello
echo $HELLO
```

Pada potongan kode tersebut tanda '\$' apa makna tanda tersebut?

Untuk mendefinisikan variabel pada bash programming.

Variabel Environment

Variabel Environment adalah variabel bawaan dari Shell itu sendiri, umumnya variabel ini berisi nilai-nilai yang merupakan konfigurasi standar dalam lingkungan program Shell yang jika digunakan akan berdampak pada system. Aturan penulisan dari environment variable adalah semua nama variabel menggunakan huruf kapital. lingkup dari variabel environment adalah bersifat GLOBAL. Variabel Environment ini juga bisa berupa dari variabel lokal yang diekspor. Untuk mengganti variabel lingkungan digunakan perintah export.

Contoh :

```
LOGNAME=shell14
SHELL=/bin/bash
HOME=/home/shell14
USER=shell14
```

#Jelaskan makna dari tiap baris dari contoh script diatas !

LOGNAME = shell14,
pencatat otomatis pada linux untuk user yang login saat ini.
SHELL = /bin/bash,
shell yang digunakan saat ini.
HOME = /home/shell14,
direktori home yang digunakan saat ini.
USER = shell14,
user yang sedang login saat ini.

```
$ env
```

```

andrian@156150600111002: ~
MANDATORY_PATH=/usr/share/gconf/ubuntu.mandatory.path
GDM_LANG=en_US
LC_MEASUREMENT=id_ID.UTF-8
IM_CONFIG_PHASE=1
COMPIZ_CONFIG_PROFILE=ubuntu
GDMSESSION=ubuntu
SESSIONTYPE=gnome-session
XDG_SEAT=seat0
HOME=/home/andrian
SHLVL=2
LANGUAGE=en_US
GNOME_DESKTOP_SESSION_ID=this-is-deprecated
LOGNAME=andrian
QT4_IM_MODULE=xim
XDG_DATA_DIRS=/usr/share/ubuntu:/usr/share/gnome:/usr/local/share:/usr/share/
DBUS_SESSION_BUS_ADDRESS=unix:abstract=/tmp/dbus-7Py3Fb5AUt
LESSOPEN=| /usr/bin/lesspipe %s
TEXTDOMAIN=im-config
INSTANCE=
DISPLAY=:0
XDG_RUNTIME_DIR=/run/user/1000
GTK_IM_MODULE=ibus
XDG_CURRENT_DESKTOP=Unity
LC_TIME=id_ID.UTF-8
LESSCLOSE=/usr/bin/lesspipe %s %s
TEXTDOMAINDIR=/usr/share/locale/
COLORTERM=gnome-terminal
XAUTHORITY=/home/andrian/.Xauthority
LC_NAME=id_ID.UTF-8
_=/usr/bin/env
andrian@156150600111002:~$ _

```

Untuk merubah nilai dari variabel environment dapat dilakukan dengan menggunakan perintah sebagai berikut :

```
$ export LOGNAME=mrbee
```

Untuk melihat hasilnya dapat dilihat dengan menggunakan perintah :

1. \$ echo \$LOGNAME
2. \$ env|grep LOGNAME

Perintah Untuk menghapus variabel environment

```
$ unset [nama_variabel]
```

```
andrian@156150600111002: ~
andrian@156150600111002:~$ echo $LOGNAME
andrian
andrian@156150600111002:~$ export LOGNAME=andrian24
andrian@156150600111002:~$ echo $LOGNAME
andrian24
andrian@156150600111002:~$
andrian@156150600111002:~$ unset LOGNAME
andrian@156150600111002:~$ echo $LOGNAME

andrian@156150600111002:~$ _
```

Variabel shell

Variabel shell adalah variabel yang ditetapkan oleh shell dan digunakan oleh shell agar berjalan dengan baik. Sebenarnya, variabel ini bisa dimasukkan dalam kategori variabel lingkungan. Contoh variabel ini adalah variabel default dari bash, misalnya:

HOME, PWD, PS1 dan PS2.

Shell juga memiliki variabel khusus yang terdiri sebuah karakter tunggal. Karakter tunggal yang memiliki makna dalam lingkungan Shell tersebut adalah :

Variabel	Arti
\$	PID dari Shell
-	Option sh yang sedang aktif
?	Status exit value untuk perintah yang terakhir dijalankan
!	PID proses terakhir yang berjalan di background

Variabel read-only adalah variabel yang mempunyai atribut read-only, artinya variabel itu tidak bisa diganti nilainya. Bahkan sebuah variabel tidak bisa dihapus dengan perintah unset jika sebuah variabel diberi atribut read-only. Contoh dari variabel read-only :

```
$word=Sisop
$readonly word
$echo $person
Sisop
$word=Praktikum
word: is read only
$
```

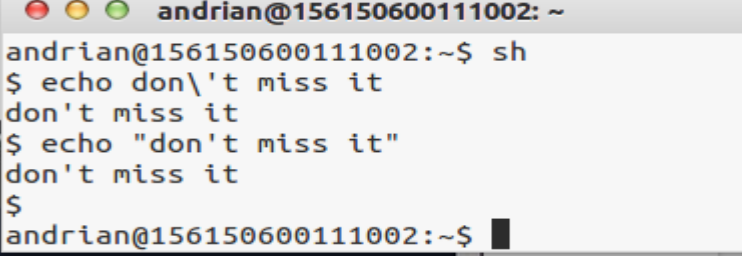
1.4.2 Quoting

Quoting adalah mekanisme untuk melindungi metakarakter dari interpretasi sebagai sebuah simbol. Shell juga mempunyai beberapa karakter yang difungsikan untuk melindungi metakarakter agar tetap diinterpretasikan sebagai karakter biasa. Ada tiga karakter quoting dalam Shell, yaitu :

1. Backslash (\)
2. Petik tunggal (')
3. Petik ganda (")

#Contoh quoting dalam Shell dan berikan keterangan!

Contoh quoting dalam Shell :



```
andrian@156150600111002: ~  
andrian@156150600111002:~$ sh  
$ echo don\'t miss it  
don't miss it  
$ echo "don't miss it"  
don't miss it  
$  
andrian@156150600111002:~$
```

Keterangan :

1. Tanda \ merupakan karakter yang meloloskan interpretasi tanda ' yang termasuk metakarakter dalam Shell, atau tanda ' yang mengikuti bukan sebuah metakarakter
2. Penggunaan tanda petik double " juga berfungsi melindungi interpretasi karakter ' sebagai metakarakter.

1.4.3 Metakarakter dalam Shell

Metakarakter adalah sebuah karakter yang memiliki arti tertentu. Dalam Shell juga dikenal beberapa metakarakter. Karena metakarakter juga ada dalam Shell maka yang perlu diperhatikan adalah kesalahan dalam penanganan sebuah karakter. Dalam sebuah kasus mencetak sebuah string di layar monitor, terkadang terjadi kasus dimana dari salah satu karakter dalam string tersebut merupakan metakarakter. Karena mengandung metakarakter maka Shell akan menginterpretasikan string tersebut tidak seperti yang diharapkan.

Berikan Contoh kasus dan jelaskan!

```
$ echo don't miss it
```

Keterangan :

Tanda ' diinterpretasikan sebagai serangkaian string sehingga Shell akan menunggu sampai tanda ' berikutnya untuk berhenti dan kemudian menampilkannya.

Jika maksudnya adalah untuk mencetak string #don#t miss it# maka yang

perlu diperhatikan adalah

```
$ echo don\'t miss it  
don't miss it
```

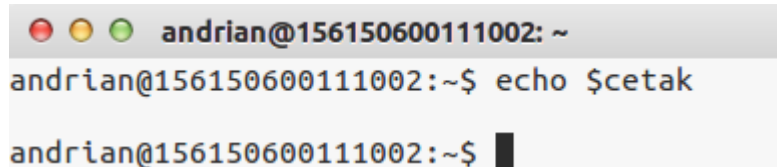
Keterangan :

1. \ merupakan karakter yang meloloskan interpretasi tanda ' yang merupakan metakarakter dalam Shell.
2. Karakter \ dikenal dengan istilah quoting dalam Shell

1.4.4 Perintah Echo

Echo biasa disebut “escape sequences character” adalah perintah untuk menampilkan data yang ada pada argumen ke standard output (stdout), yang dalam hal ini stdout bisa merupakan layar monitor atau juga sebuah file. Perintah Echo dalam Shell memiliki opsi-opsi untuk membentuk atau memberikan format pada data yang dikeluarkan.

```
$ echo $cetak
```



The screenshot shows a terminal window with the prompt 'andrian@156150600111002: ~'. The user enters the command 'echo \$cetak', and the terminal displays the output 'andrian@156150600111002:~\$ echo \$cetak' followed by a new line.

1.4.5 Fungsi

Fungsi adalah program yang dapat dipanggil berulang-ulang oleh program lainnya dengan menggunakan notasi NamaFungsi(). Di dalam Shell fungsi juga bisa didefinisikan interaktif maupun secara skrip program, dan meskipun didefinisikan secara interaktif, sebuah fungsi juga bisa dipanggil melalui skrip yang dibuat dalam sebuah file dengan catatan fungsi tersebut sudah di export. Setelah melalui mekanisme export ini sub-shell juga bisa memanggil fungsi tersebut.

Bentuk umum dalam mendefinisikan fungsi dalam BASH Shell adalah sebagai berikut :

```
Nama_fungsi () { command; command; }  
function nama_fungsi { command;command; }  
function nama_fungsi () { command; command; }
```

1.4.6 Array

Pada versi BASH 2.x terdapat fungsi untuk mendefinisikan array satu dimensi. Array memungkinkan seorang programmer mengkoleksi daftar beberapa nilai dalam sebuah variabel. Untuk mengekstraksi kembali nilai-nilai tersebut dapat

dilakukan dengan menyebutkan nama variabel yang diikuti oleh nomer indek array tersebut.

Pendefinisian sebuah array juga bisa dilakukan on the fly(tanpa mendefinisikan terlebih dahulu). Dan tidak ada batasan maksimum dari sebuah array yang dibuat dalam lingkungan BASH Shell. Pada saat sebuah nilai diberikan ke dalam sebuah array yang telah didefinisikan, indek array secara otomatis akan dimulai dari 0, dan bertambah naik 1 sampai semua kumpulan nilai-nilai dimasukkan.

#Sebutkan dan jelaskan macam-macam cara untuk mendeklaraasikan array!

Dengan mengisi langsung elemen - elemen array dengan tanda kurung tanpa mendeklarasikan elemen tersebut berada di index ke berapa.

Contoh:

```
buah={Apel, Melon, Durian};
```

Di sini bash akan secara otomatis menempatkan melon di index ke 0, apel di index ke 1, dan durian di index ke 2. dengan

Contoh lain saya menggunakan array hewan.

```
hewan [0] = 0
```

```
hewan [1] = harimau
```

2. Percobaan

Cobalah sebagai latihan shell programming.

*Untuk percobaan 1, kita gunakan **Shell Interaktif***

Percobaan 1

- Percobaan 1 adalah shell interaktif dengan menggunakan fungsi if

```
$ if grep "root" /etc/passwd
> then
> echo "disini ada user yang bernama root"
> else
> echo "user tersebut tidak ada"
> fi
```

*Untuk percobaan 2 dan seterusnya, kita gunakan akses **file.sh***

Percobaan 2

- Percobaan 2 s.d. 4 adalah mengakses variabel shell

```
#!/bin/bash
echo "selamat datang $USER"
echo "di shell programming"
```

#Jelaskan arti dari \$USER dan apa yang terjadi tanpa menggunakan '\$' !

Dengan \$, user yang menjalankan perintah ini akan ditampilkan, sementara tanpa \$, hanya akan tampil kata "USER".

Percobaan 3

```
#!/bin/bash
echo "Shell yang digunakan adalah $SHELL"
echo "saat ini jam `date +%T`"
echo "tanggal `date +%D`"
```



The screenshot shows a terminal window with the title bar "andrian@156150600111002: ~". The terminal content is as follows:

```
andrian@156150600111002:~$ sh 2.sh
"selamat datang andrian"
"di shell programming"
andrian@156150600111002:~$
andrian@156150600111002:~$ sh 2.sh
"selamat datang USER"
"di shell programming"
andrian@156150600111002:~$ _
```

#coba tampilkan semua text dengan warna merah !

Untuk mengubah menjadi warna merah,
tambahkan echo -e di awal dan akhir code.

```
andrian@156150600111002: ~  
GNU nano 2.2.6 File: file.sh  
#!/bin/bash  
echo -e "\033[31m"  
echo "Shell yang digunakan adalah $SHELL"  
echo "saat ini jam `date +%T`"  
echo "tanggal `date +%D`"  
echo -e "\033[0m"  
  
andrian@156150600111002: ~  
andrian@156150600111002:~$ nano file.sh  
andrian@156150600111002:~$ bash file.sh  
  
Shell yang digunakan adalah /bin/bash  
saat ini jam 11:48:15  
tanggal 04/19/16  
  
andrian@156150600111002:~$
```

Percobaan 4

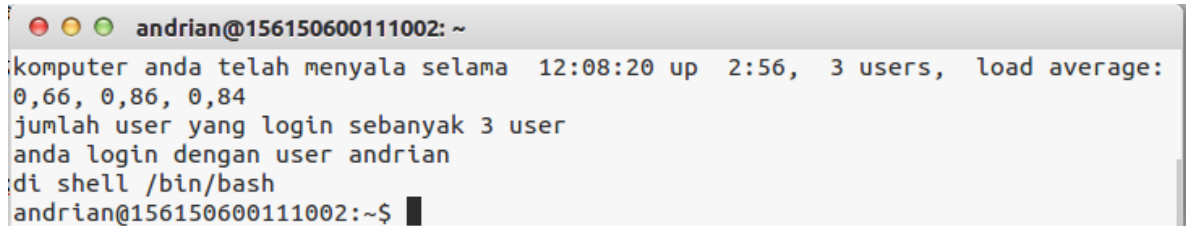
```
#!/bin/bash  
clear  
echo "nama login anda $LOGNAME"  
echo "saat ini anda berada di direktori `pwd`"  
echo "waktu sekarang adalah `date +%T`"  
echo "selamat bekerja"
```

```
andrian@156150600111002: ~  
GNU nano 2.2.6 File: file.sh  
#!/bin/bash  
clear  
echo "nama login anda $LOGNAME"  
echo "Saat ini anda berada di derktori `pwd`"  
echo "waktu sekarang `date +%T`"  
echo "selamat bekerja"
```

Percobaan 5

- Percobaan 5 terdapat metakarakter untuk perintah dasar linux

```
#!/bin/bash
clear
echo "komputer anda telah menyala selama `uptime`"
echo "jumlah user yang login sebanyak `who | wc -l`
user"
echo "anda login dengan user $LOGNAME"
echo "di shell $SHELL"
```



```
andrian@156150600111002: ~
komputer anda telah menyala selama 12:08:20 up 2:56, 3 users, load average:
0,66, 0,86, 0,84
jumlah user yang login sebanyak 3 user
anda login dengan user andrian
di shell /bin/bash
andrian@156150600111002:~$
```

#Jelaskan makna dari tiap baris script yang terdapat diatas!

- 1)Perintah `uptime` digunakan untuk melihat berapa lama komputer atau laptop kita berjalan sejak anda hidupkan
- 2)Perintah `who | wc -l` digunakan untuk melihat jumlah user yang menggunakan sistem
- 3)\$LOGNAME digunakan untuk mengakses nilai variable LOGNAME

Percobaan 6

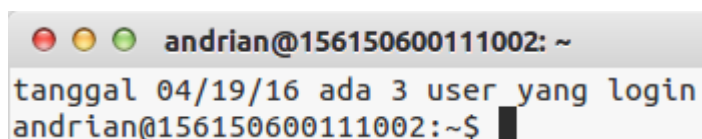
- Percobaan 6 memasukkan metakarakter perintah dasar linux kedalam variabel lokal

```
#!/bin/bash
clear
data='date +%D'
jumlah=`who | wc -l`
echo "tanggal $data ada $jumlah user yang login"
```

#Apa maksud dari perintah +%D ? jelaskan!

Memanggil fungsi tanggal

Percobaan 7



```
andrian@156150600111002: ~
tanggal 04/19/16 ada 3 user yang login
andrian@156150600111002:~$
```

- Percobaan 7 mengatur quoting

```
#!/bin/bash
user=andi
echo "hai $user i'm glad to meet you"
echo hai $user i\#m glad to meet you
echo "apakah kamu punya teman
"special"?"
echo "\"special\" ?? teman apa itu?"
```

#Identifikasikan metakarakter yang ada didalam perintah diatas!

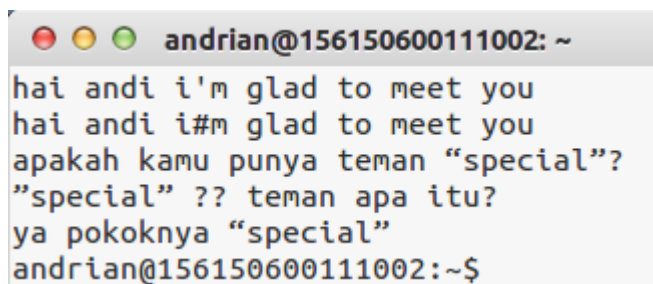
User = andi, jadi saat user yang dipanggil, maka andi yang akan ditampilkan.

\$user, menampilkan nama andi.

i\#m, menampilkan kata i#m karena menggunakan backslash.

"spesial", menampilkan spesial, karena tidak ada backslash.

"\"spesial\" ", kebalikan dari poin atasnya.

A screenshot of a terminal window with a title bar showing three colored circles (red, yellow, green) and the text "andrian@156150600111002: ~". The terminal displays the output of the script: "hai andi i'm glad to meet you", "hai andi i#m glad to meet you", "apakah kamu punya teman \"special\"?", "\"spesial\" ?? teman apa itu?", "ya pokoknya \"special\"", and the prompt "andrian@156150600111002:~\$ _".

```
andrian@156150600111002: ~
hai andi i'm glad to meet you
hai andi i#m glad to meet you
apakah kamu punya teman "special"?
"spesial" ?? teman apa itu?
ya pokoknya "special"
andrian@156150600111002:~$ _
```

Percobaan 8

- Percobaan 8 membuat fungsi

```
#!/bin/bash
function cetak_selamat {
echo "Selamat Datang"
echo "Di Shell Programming"
}
cetak_selamat
```

#apa hasil keluaran dari fungsi diatas dan berikan contoh fungsi yang lain!

```
andrian@156150600111002: ~  
GNU nano 2.2.6 File: file.sh  
  
clear  
function cetak_selamat {  
echo "Selamat Datang"  
echo "Di Shell Programming"  
} cetak_selamat  
  
andrian@156150600111002: ~  
Selamat Datang  
Di Shell Programming  
andrian@156150600111002:~$
```

```
andrian@156150600111002: ~  
GNU nano 2.2.6 File: file.sh  
  
clear  
function cetak_selamat {  
echo "Selamat Datang $USER"  
echo "Di Shell Programming"  
echo "ini hanya tulisan"  
}  
cetak_selamat  
  
andrian@156150600111002: ~  
Selamat Datang andrian  
Di Shell Programming  
ini hanya tulisan  
andrian@156150600111002:~$
```

Percobaan 9

- Percobaan 9 variasi pemanggilan variabel

```
#!/bin/bash  
data="halo ini bejo"  
sdata=${data#*lo}  
tdata=${data%be*}  
echo $sdata  
echo $tdata
```

#screenshot dan analisis dari hasil output diatas!



```
andrian@156150600111002: ~  
ini bejo  
halo ini  
andrian@156150600111002:~$
```

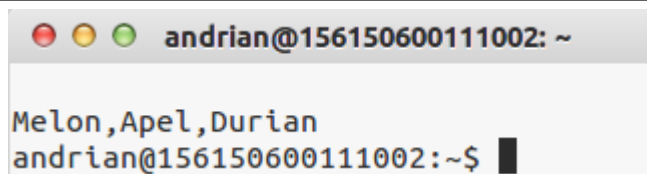
1. `${data#*lo}` = menampilkan teks setelah kata "lo". Jika pada variable data kata "lo" adl ini bejo shg akan menampilkan " ini bejo".
2. `${data%be*}` = kata berawaln "be" dan sterusnya tdk akan ditampilkan. Disini kata berwalan "be" adalah bejo maka tidak ditampilkan sehingga hanya menampilkan "halo ini".

Percobaan 10

- Percobaan 10 membuat array

```
#!/bin/bash  
#array  
buah=(Melon,Apel,Durian);  
echo ${buah[*]};
```

#jelaskan arti dari fungsi [*] diatas dan analisis outputnya!



```
andrian@156150600111002: ~  
Melon,Apel,Durian  
andrian@156150600111002:~$
```

Menampilkan semua isi data array, output menghasilkan "Melon, Apel, Durian".

Jalankan program dibawah ini!

```

while :
do
    clear
    echo "-----"
    echo " Main Menu "
    echo "-----"
    echo "[1] Show Todays date/time"
    echo "[2] Show files in current directory"
    echo "[3] Show calendar"
    echo "[4] Start editor to write letters"
    echo "[5] Exit/Stop"
    echo "======"
    echo -n "Enter your menu choice [1-5]: "
    read yourch
    case $yourch in
        1) echo "Today is date , press a key. . ." ; read ;;
        2) echo "Files in pwd" ; la; echo "Press a key. . ." ;
read ;;
        3) cal ; echo "Press a key. . ." ; read ;;
        4) vi ;;
        5) exit 0 ;;
        *) echo "Oops!!! Please select choice 1,2,3,4, or 5";
echo "Press a key. . ." ; read ;;
    esac
done

```

#Dalam program diatas masih banyak kesalahan, jelaskan dan benahi program tersebut!

```
andrian@156150600111002: ~
GNU nano 2.2.6 File: file.sh

clear
yourch=0
while :
do
    clear
    echo "-----"
    echo " Main Menu "
    echo "-----"
    echo "[1] Show Todays date/time"
    echo "[2] Show files in current directory"
    echo "[3] Show calendar"
    echo "[4] Start editor to write letters"
    echo "[5] Exit/Stop"
    echo "===== "
    echo -n "Enter your menu choice [1-5]: "
    read yourch
    case $yourch in
        1) echo "Today is `date` , press a key. . ." ; read ;;
        2) echo "Files in `ls -la` , Press a key. . ." ; read ;;
        3) cal ; echo "Press a key. . ." ; read ;;
        4) nano;;
        5) exit 0;;
        *) echo "Oops!!! Please select choice 1,2,3,4, or 5"; echo "Press a key. . ." ; read ;;
    esac
done
```

Program diatas menggunakan while untuk looping dari pernyataan-pernyataan yang ada . kemudian juga terdapat perintah do untuk memulai perulangan, dan di bawahnya ada perintah clear untuk menghapus tampilan dari terminal (shell). Setelah perintah clear barulah dikeluarkan pilihan yang ditampilkan ke user dan meminta masukan (input) kepada user sesuai no pilihan yang tersedia dengan perintah read dan menyimpannya ke dalam variabel yourch. Dan setelah selesai dengan statement case, ditutup dengan statement esac kemudian menutup perulangan while-do dengan statement done sebagai akhir dari perulangan.

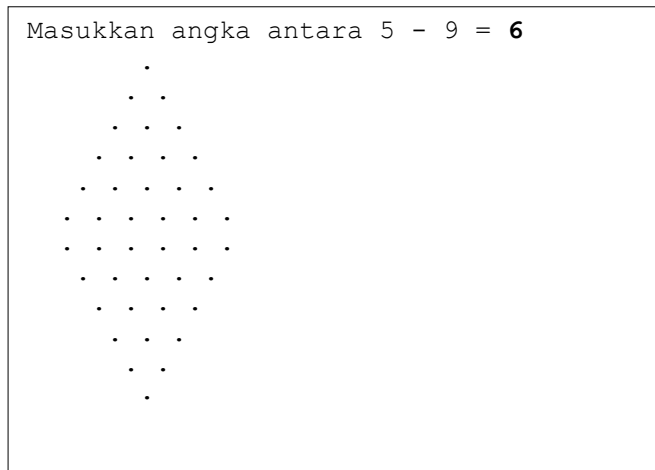


LAPORAN PRAKTIKUM SISTEM OPERASI FAKULTAS ILMU KOMPUTER UNIVERSITAS BRAWIJAYA

Nama : Moh. Arif Andrian
NIM : 156150600111002
Tugas : BAB V
Asisten : Siska Permatasari
Zaenal Kurniawan

TUGAS PRAKTIKUM

1. Dengan menggunakan nilai input antara 5 sampai dengan 9, Tampilkan



output seperti berikut ini: (2 segitiga titik saling bersinggungan)

Jawab:

```
root@156150600111002: /home/andrian
GNU nano 2.2.6      File: soal1.sh

#!/bin/bash
angka=0
echo -n "Masukkan angka antara 5 - 9 = "
read angka
if ! [ $angka -ge 5 -a $angka -le 9 ] ; then
    echo "Opps!!! mohon masukkan angka 5,6,7,8, atau 9"
    exit 1
fi
##### piramid keatas #####
for (( i=1; i<=angka; i++ ))
do
    for (( s=angka; s>=i; s-- ))
    do
        echo -n " "
    done
    for (( j=1; j<=i; j++ ))
    do
        echo -n " ."
    done
    echo ""
done
##### piramid kebawah #####
for (( i=angka; i>=1; i-- ))
do
    for (( s=i; s<=angka; s++ ))
    do
        echo -n " "
    done
    for (( j=1; j<=i; j++ ))
    do
        echo -n " ."
    done
    echo ""
done
done
```

```
root@156150600111002: /home/andrian
root@156150600111002:/home/andrian# bash soal1.sh
Masukkan angka antara 5 - 9 = 6

      .
     . .
    . . .
   . . . .
  . . . . .
 . . . . . .
. . . . . . .
 . . . . .
  . . . .
   . . .
    . .
     .
      .

root@156150600111002:/home/andrian#
```

2. Bagaimana melakukan proses debug sebuah script shell?

Jawab:

penggunaan perintah **echo**.

Jalankan script dengan parameter **-x**.

Misalnya, `sh -x./myscript`

atau **set xtrace -o** sebelum menjalankan script.

Pilihan ini dapat ditambahkan ke baris pertama dari script di mana shell didefinisikan, misalnya `#!/bin/sh -xv`

3. Buatlah perintah untuk menghitung perkalian aritmatika dengan menggunakan variabel inputan?

Jawab:

```
root@156150600111002: /home/andrian
GNU nano 2.2.6 File: soal3.sh

#!/bin/bash
echo "OPERASI PERKALIAN ARITMATIKA"

echo -n "Masukan nilai ke 1 : "
read nilai1

echo -n "Masukan nilai ke 2 : "
read nilai2

let "hasil = $nilai1 * $nilai2";

echo "Hasil $nilai1 x $nilai2 = $hasil";
```

```
root@156150600111002: /home/andrian
root@156150600111002:/home/andrian# nano soal3.sh
root@156150600111002:/home/andrian# bash soal3.sh
OPERASI PERKALIAN ARITMATIKA
Masukan nilai ke 1 : 4
Masukan nilai ke 2 : 6
Hasil 4 x 6 = 24
```



LAPORAN PRAKTIKUM SISTEM OPERASI

FAKULTAS ILMU KOMPUTER

UNIVERSITAS BRAWIJAYA

Nama : Moh. Arif Andrian
NIM : 156150600111002
Kesimpulan : Bab V
Asisten : Siska Permatasari
Zaenal Kurniawan

KESIMPULAN

Shell adalah program yang menghubungkan user dengan sistem operasi dalam hal ini kernel (inti sistem operasi), umumnya shell menyediakan prompt sebagai user interface, tempat di mana user menginputkan perintah-perintah yang diinginkan baik berupa perintah internal shell (internal command), ataupun perintah eksekusi suatu file program (eksternal command), selain itu shell memungkinkan user menyusun sekumpulan perintah pada sebuah atau beberapa file untuk dieksekusi sebagai program.

Tidak seperti sistem operasi lain yang hanya menyediakan 1 atau 2 shell, sistem operasi dari keluarga unix misalnya linux sampai saat ini dilengkapi oleh banyak shell dengan kumpulan perintah yang sangat banyak, sehingga memungkinkan pemakai memilih shell mana yang paling baik untuk membantu menyelesaikan pekerjaannya, atau dapat pula berpindah-pindah dari shell yang satu ke shell yang lain dengan mudah,

Beberapa contoh shell yang ada di linux, antara lain:

- Bourne shell(sh),
- C shell(csh),
- Korn shell(ksh),
- Bourne again shell(bash).

Masing – masing shell mempunyai kelebihan dan kekurangan yang mungkin lebih didasarkan pada kebutuhan pemakai yang makin hari makin meningkat, untuk dokumentasi ini shell yang digunakan adalah bash shell dari GNU, yang merupakan pengembangan dari Bourne shell dan mengambil beberapa fitur (keistimewaan) dari C shell serta Korn shell, Bash shell merupakan shell yang cukup banyak digunakan pemakai linux karena kemudahan serta banyaknya fasilitas perintah yang disediakan.