**Laporan Pratikum**

**Sistem Operasi Dasar**

PRATIKUM 10



Dosen Pengampu : Ibnu Surya, S.T.,M.T.

AIL : Iqbal Mahatma Putra S.S.T

Nama  : Laurin Madelau

NIM    : 2055301067

Kelas    : 1 TI C

**JURUSAN TEKNIK INFORMATIKA**

**POLITEKNIK CALTEX RIAU**

**2021**

PERCOBAAN

Percobaan 1: Membuat shell script

1. Buatlah file prog01.sh dengan editor vi

$ vi prog01.sh

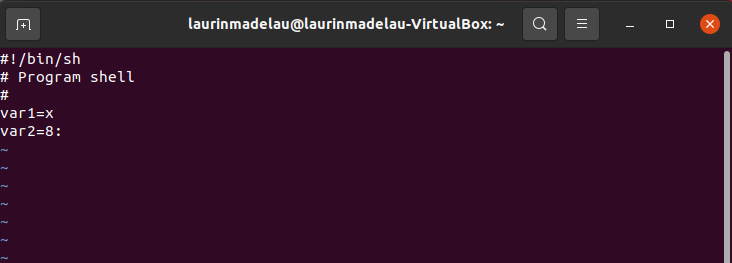
#!/bin/sh

# Program shell

#

var1=x

var2=8



* Gambar di atas merupakan perintah untuk membuat shell script dengan membuat file prog01.sh menggunakan editor vi. Perintah $ vi prog01.sh berfungsi untuk mengedit file konfigurasi sistem dan membuat script. Kemudian tekan i untuk insert script. Script diatas berfungsi untuk mendifinisikan var1 sama dengan x dan var2 sama dengan 8. tekan esc pada keyboard lalu setelah tulisan INSERTnya hilang ketikan :wq berfungsi untuk menyimpan dan keluar dari file tersebut.

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

$ . prog01.sh



* Gambar di atas merupakan perintah untuk menjalankan shell, gunakan notasi TITIK di depan nama program. Ketikkan perintah di atas, tapi prog01.sh ini tidak menampilkan teks/string seperti menggunakan perintah echo, maka pada saat prog01.sh dijalankan tidak akan ditampilkan apa-apa.

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

$ chmod +x prog01.sh

$ ./prog01.sh



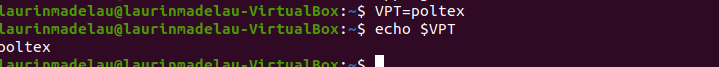
* tidak akan menampilkan apa-apa karena isi dari script prog01.sh hanya mendefiniskan var1 dan var2 tanpa ada proses menampilkan.

Percobaan 2: Variabel

1. Contoh menggunakan variabel pada shell interaktif

$ VPT=poltex

$ echo $VPT



* Gambar di atas merupakan contoh menggunakan variable pada shell interaktif. Pada shell aktif untuk menampilkan variabel pertama definiskan nama variabel tersebut, misalnya variabel VPT berisi poltek. Dan untuk menampilkan atau memanggil variabel tersebut dapat menggunakan echo $VPT maka setelah dienter isi dari variabel VPT akan tampil yaitu output dengan tulisan poltek.

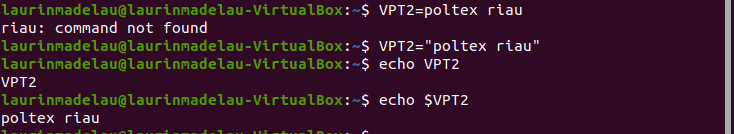
2. Pemisahan 2 kata dengan spasi menandakan eksekusi 2 buah instruksi. Karakter $ harus ada pada awal nama variabel untuk melihat isi variabel tersebut, jika tidak, maka echo akan mengambil parameter tersebut sebagai string.

$ VPT2=poltex riau (Terdapat pesan error)

$ VPT2=”poltex riau”

$ echo VPT2

$ echo $VPT2



* Gambar di atas merupakan pemisah 2 kata dengan spasi menandakan eksekusi 2 buah intruksi. Terdapat beberapa perbedaan utnuk program di atas. Untuk isi dari sebuah variabel apabila terdiri dari dua buah kata maka tidak bisa secara langsung dipisahkan dengan spasi jadi terdapat pesan error. Untuk membuat variabel dengan 2 kata harus diaapit oleh tanda kutip dua (“). Kemudian pada perintah selanjutnya untuk menampilkan variabel VPT2 ini harus diawali dengan tanda $ jika tidak, maka echo akan mengambil parameter tersebut sebagai string.

3. Menggabungkan dua variabel atau lebih

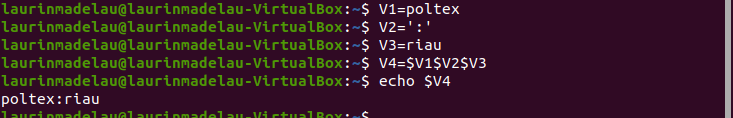
$ V1=poltex

$ V2=’:’

$ V3=riau

$ V4=$V1$V2$V3

$ echo $V4



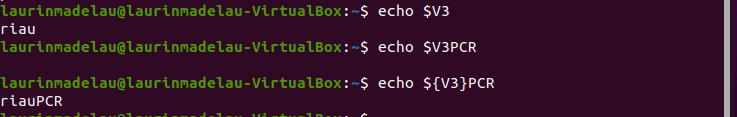
* Gambar di atas merupakan penggabungan dua variable atau lebih. Untuk menggabungkan dua variabel atau lebih selain dapat ditulis secara langsung bisa juga didefinisikan lagi kedalam variabel lain contohnya pada gambar diatas tiga buah variabel didefiniskan di variabel V4 maka untuk menampilkan variabel V1 V2 V3 cukup memanggil variabel V4 saja.

4. Menggabungkan isi variabel dengan string yang lain. Jika digabungkan dengan nama variabel yang belum didefinisikan (kosong) maka instruksi echo menghasilkan string kosong. Untuk menghindari kekeliruan, nama variabel perlu diproteksi dengan {} dan kemudian isi variabel tersebut digabung dengan string.

$ echo $V3

$ echo $V3PCR

$ echo ${V3}PCR



* Gambar di atas merupakan penggabungan isi variable dengan string yang lain. Untuk menggabungkan sebuah variabel dengan string lain tidak dapat secara langsung seperi pada perintah 2 diatas. Jika penulisan nya seperti $V3PCR maka instruksi echo akan menampilkan string kosong karena variabel tersebut dianggap sebagai variabel yang belum terdefinisikan. Untuk menggabungkan isi variabel dengan string lain digunakan tanda {} untuk memisahkan variabel dengan string lain tersebut.

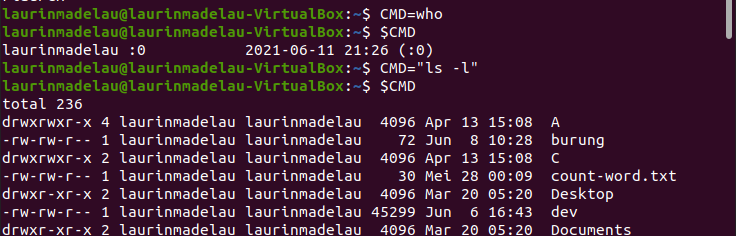
5. Variabel dapat berisi instruksi, yang kemudian bila dijadikan input untuk shell, instruksi tersebut akan dieksekusi

$ CMD=who

$ $CMD

$ CMD=”ls –l”

$ $CMD



* Gambar di atas merupakan variabel berisi instruksi, yang kemudian bila dijadikan instruksi tersebut akan dieksekusi. Variabel dapat berisi isntruksi. Seperti pada contoh diatas variabel CMD diisi oleh instruksi who, maka apabila variabel CMD dipanggil isntruksi who yang tedapat dalam variabel CMD akan dieksekusi. Begitu juga dengan variabel CMD yang disi dengan perintah ls –l, dan tampil dengan output menampilkan file dile pada direktori.

6. Modifikasi file prog01.sh berikut

$ vi prog01.sh

#!/bin/sh

V1=poltex

V2=’:’

V3=riau

echo “Pemrograman shell”

echo $V1$V2$V3

V3=PCR

echo $V1$V2 di $V3



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

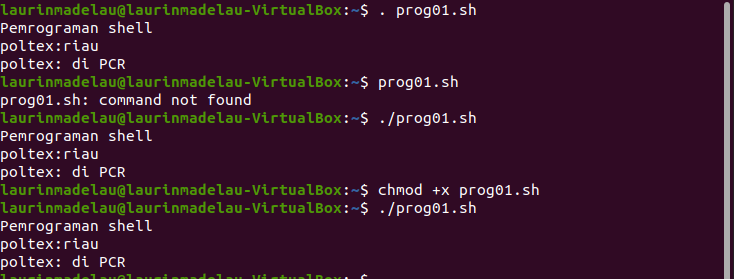
$ . prog01.sh

$ prog01.sh (Terdapat pesan error)

$ ./prog01.sh (Terdapat pesan error)

$ chmod +x prog01.sh

$ ./prog01.sh



* Gambar di atas merupakan cara untuk menjalankan prog01.sh. Untuk memanggil prog01.sh ini dapat dilakukan dengan beberapa cara seperti diatas. Pemanggilan kedua dan ketiga gagal, karena pemanggilan tersebut hanya untuk file execute.

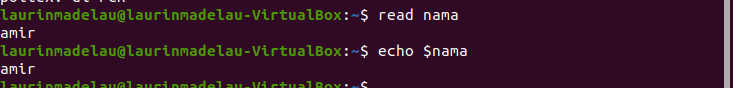
Percobaan 3: Membaca keyboard

1. Menggunakan instruksi read

$ read nama

amir

$ echo $nama



* Gambar di atas merupakan cara untuk membaca keyboard. Untuk membaca input dari keyboard dapat menggunakan instruksi read. Program diatas variabel nama akan dinput melalui keyboard yaitu amir, pada saat dipanggil maka variabel nama tersebut kan menampilkan amir.

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 : “

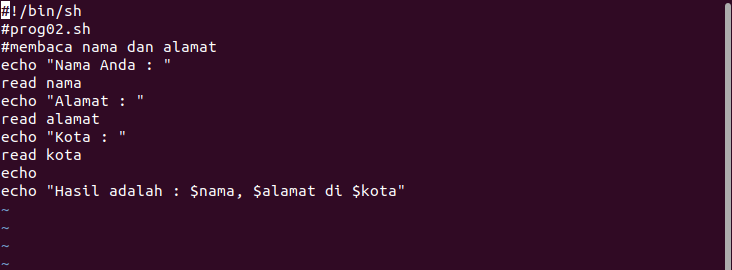
read alamat

echo “Kota : “

read kota

echo

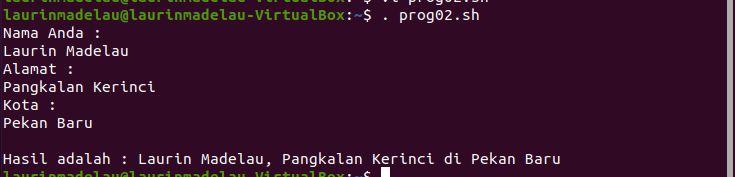
echo “Hasil adalah : $nama, $alamat di $kota”



* Gambar di atas merupakan pembuatan file prog02.sh. Sama halnya dengan percobaan pada point sebelumnya, namun percobaan membaca variabel dari keyboard ini dilakukan melalui shell script prog02.sh. 3 buah variabel nama alamat dan kota merupakan input dari keyboard menggunkan perintah read dan akan di eksekusi pada percobaan berikutnya.

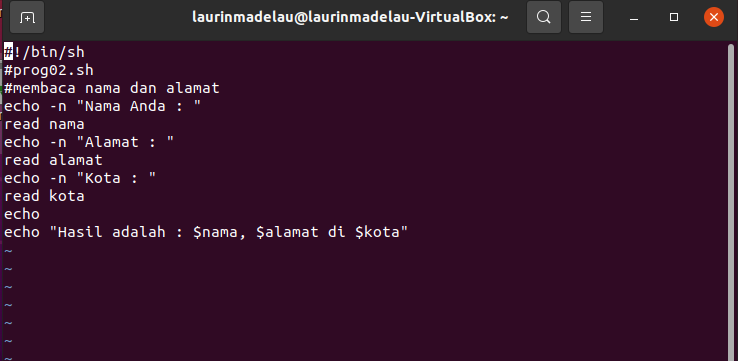
3. Eksekusi program prog02.sh

$ . prog02.sh



* Gambar di atas merupakan perintah untuk menjalankan proses membaca keyboard. Pada saat prog02.sh dieksekusi akan tampil tulisan nama anda : kemudian diikuti dengan perintah memasukan input dari variabel nama dengan posisi satu enter. Begitu juga untuk variabel alamat dan kota. Kemudian setelah menginputkan variabel kota maka akan tampil Hasil adalah : dengan variabel nama, varibael alamat dan diikuti di dengan variabel kota.

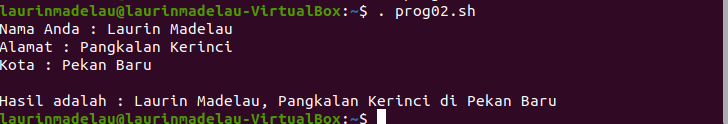
4. Instruksi echo secara otomatis memberikan baris baru, maka untuk menghindari hal tersebut disediakan opsi –n, yang menyatakan kepada echo untuk menghilangkan baris baru. Modifikasi program prog02.sh



* Gambar di atas merupakan perintah untuk memodifikasi prog02.sh. Pada prog02.sh yang baru ini perintah echo diikuti –n untuk mengilangkan baris baru untuk melakukan input keyboard dari masing masing variabel dan akan di eksekusi pada percobaan berikutnya.

5. Eksekusi program prog02.sh

$ . prog02.sh



* Gambar di atas merupakan perintah untuk membaca keyboard. Prog02.sh tersebut di eksekusi sehingga tampil dan terlihat perbedaan dengan percobaan 3 nomer 3 untuk input keyboard tidak dilakukan pada baris baru namun meneruskan dari perintah echo.

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

$ read nama

<CR>

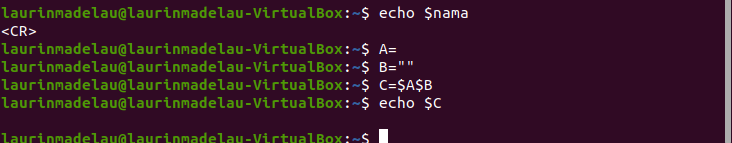
$ echo $nama

$ A=

$ B=””

$ C=$A$B

$ echo $C



* Gambar di atas merupakan perintah untuk membaca keyboard. Seluruh perintah dari variabel diatas merupakan variabel kosong atau tidak mempunyai nilai. Maka pada saat variabel dipanggil tidak akan menamilkan apapun.

7. Variabel dapat disubtitusikan dengan hasil eksekusi dari sebuah

instruksi. Pada contoh dibawah, instruksi pwd dieksekusi lebih dahulu

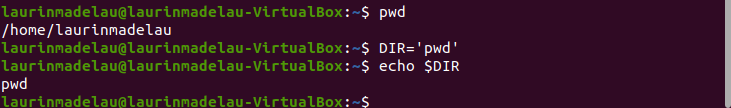
dengan sepasang Back Quate (tanda kutip terbalik). Hasil dari

eksekusi tersebut akan masuk sebagai nilai variabel DIR

$ pwd

$ DIR=`pwd`

$ echo $DIR



* Gambar di atas merupakan perintah untuk membaca keyboard. Untuk mengisi variabel dengan subtitusi hasil eksekusi dari sebuah eksekusi instruksi harus menggunakan sepasang back quate (`). Misalnya pada contoh diatas variabel DIR didistribusikan dengan hasil eksekusi dari peirntah pwd. Maka saat variabel DIR dipanggil akan tampil hasil dari pwd tersebut.

8. Buatlah shell script prog03.sh

$ vi prog03.sh

#!/bin/sh

# prog03.sh

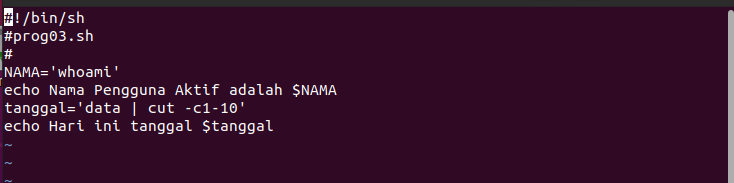
#

NAMA=`whoami`

echo Nama Pengguna Aktif adalah $NAMA

tanggal=`date | cut –c1-10`

echo Hari ini tanggal $tanggal



9. Eksekusi prog03.sh

$ . prog03.sh



* Gambar di atas merupakan hasil dari eksekusi shell script prog03.sh. nama pengguna aktif berisi variabel NAMA yaitu hasil eksekusi dari perintah whoami. Kemudian Hari ini tanggal akan diikuti dengan variabel tanggal dengan isi hasil dari instruksi date.

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”

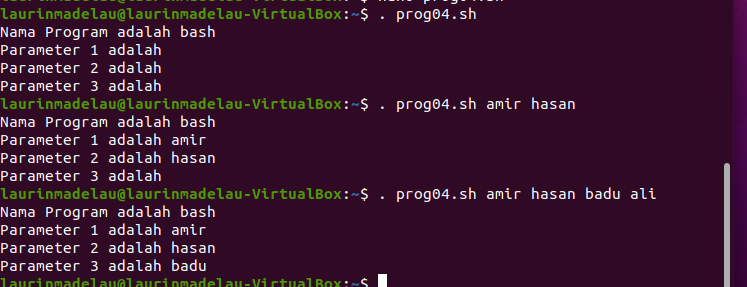


2. Eksekusi prog04.sh tanpa parameter, dengan 2 parameter, dengan 4 parameter

$ . prog04.sh

$ . prog04.sh amir hasan

$ . prog04.sh amir hasan badu ali



* Gambar di atas merupakan perintah membaca parameter. Apabila prog04.sh hanya di eksekusi tanpa diikuti dengan parameter maka akan menampilkan nama program dengan $0 adalah bash sedangkan parameter 1 2 dan 3 tidak akan berisi parameter. Eksekusi prog04.sh kedua dan ketiga diikuti dengan beberapa parameter. Pertama hanya ada satu parameter yaitu badu maka parameter yang akan diisi hanya parameter 1. Sedangkan selanjutnya diiktui dengan 3 buah parameter

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”

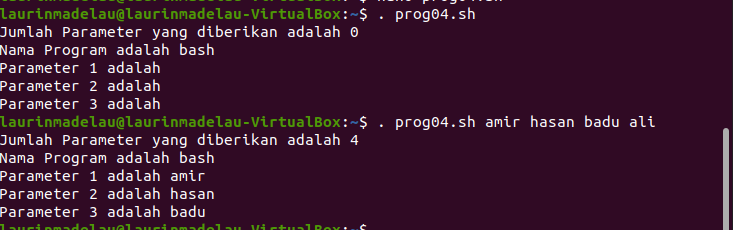
echo “Parameter 3 adalah $3”



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

$ . prog04.sh

$ . prog04.sh amir hasan badu ali



* Gambar di atas merupakan perintah membaca parameter. Eksekusi pertama prog04.sh tidak diisi dengan parameter maka $# akan menampilkan 0, sedangkan eksekusi porg04.sh diikuti dengan badu coba lagi dihitung sebagai 4 parameter, dan tampil dengan jumlah parameternya ditampilkan.

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”

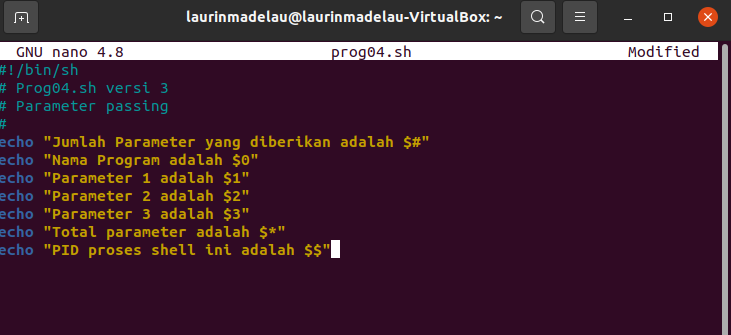
echo “Parameter 1 adalah $1”

echo “Parameter 2 adalah $2”

echo “Parameter 3 adalah $3”

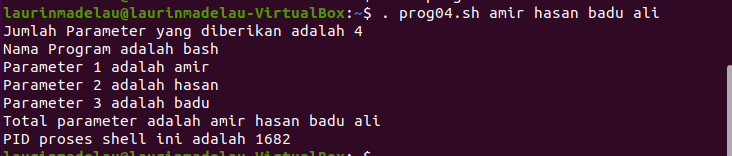
echo “Total parameter adalah $\*”

echo “PID proses shell ini adalah $$”



6. Eksekusi prog04.sh dengan 4 parameter

$ . prog04.sh amir hasan badu ali



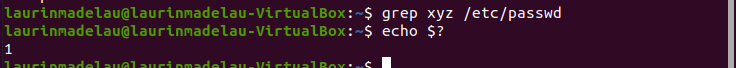
* Gambar di atas merupakan perintah membaca parameter. Total parameter adalah merupakah hasil untuk menampilkan damelia coba lagi dari instruksi $\*. Sedangakn PID proses shell ini adalah 1682 dengan instruksi $$, dan menampilkan program sebelumnya dengan menampilkan juga no PID yang digunakan.

Percobaan 5: Status exit

1. String tidak diketemukan, maka status exit adalah 1

$ grep xyz /etc/passwd

$ echo $?

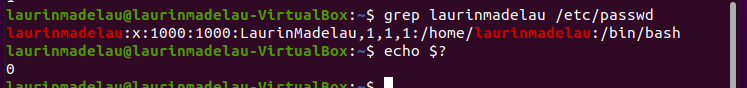


* Gambar di atas merupakan perintah status exit. Apabila string yang tidak ditemukan pada instruksi grep seperti diatas, akan ditampilkan status exit dengan perintah echo $? adalah 1, artinya tidak ditemukan.

2. String diketemukan, maka status exit adalah 0

$ grep studentX /etc/passwd

$ echo $?



* Gambar di atas merupakan perintah status exit. Pada contoh diatas ini string laurinmadelau ditemukan dalam directory /etc/passwd maka apabila echo $? dipanggil akan menampilkan status exit 0, dan artinya string ditemukan.

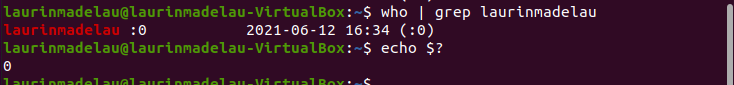
Percobaan 6: Konstruksi if

1. Instruksi dengan exit status 0

$ who

$ who | grep studentX

$ echo $?



* Gambar di atas merupakan perintah konstruksi if. Masih merupakan bahasan status exit dari string laurinmadelau dari instruksi who, maka status exitnya adalah 0 dan tampil dengan menampilkan hasil yang berkaitan dengan perintah who.

2. if membandingkan exit status dengan 0, bila sama, maka blok

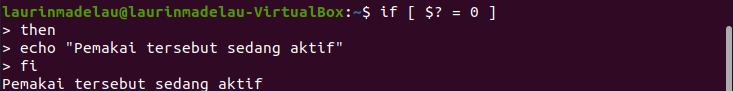
program masuk ke dalam blok then-fi

$ if [ $? = 0 ]

> then

> echo “Pemakai tersebut sedang aktif”

> fi



* Gambar di atas merupakan perintah konstruksi if. Perintah if ini adalah untuk membandingkan status exit dengan 0, bila sama, maka program masuk ke dalam blok then if yaitu Pemakai tersebut sedang aktif, karena perintah sebelumnya user damelia sedang aktif hasilnya 0, apabila dengan if $? = 1 artinya user tidak aktif.

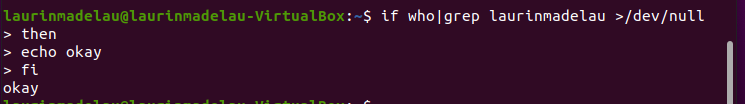
3. Nomor (1) dan (2) di atas dapat disederhanakan dengan

$ if who|grep studentX >/dev/null

> then

> echo okay

> fi



* Gambar di atas merupakan perintah konstruksi if. Pada percobaan ini merupakan penyederhanaan dari nomor 1 dan 2. Dengan instruksi if who dengan grep damelia akan dibelokan ke /dev/null maka apabila perintahnya berstatus exit 0 akan ditampilkan okay.

Percobaan7 : Konstruksi if then else

1. Membuat shell script prog05.sh

$ vi prog05.sh

#!/bin/sh

# prog05.sh

# Program akan memberikan konfirmasi 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

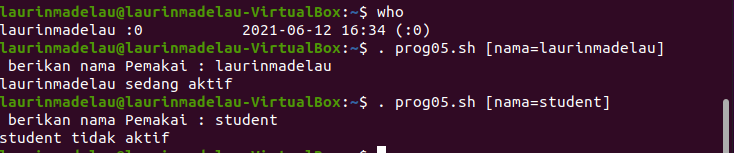
tampil pada instruksi who dan coba juga untuk nama pemakai yang

tidak aktif

$ who

$ . prog05.sh [nama=studentX]

$ . prog05.sh [nama=student]



* Gambar di atas merupakan perintah menampilkan konstruksi if then else. Perintah who ini menampilkan user aktif yaitu laurinmadelau dan user yang tidak aktif yaitu student. Jika prog05.sh diisi dengan laurinmadelau, maka akan ditampilkan laurinmadelau sebagai variabel nama yaitu laurinmadelau sedang aktif. Sedangkan jika prog05.sh diisi dengan nama pemakai student maka instruksi ini dinyatakan tidak aktif karena user yang aktif adalah laurinmadelau.

Percobaan 8: Instruksi test

1. Menggunakan instruksi test, perhatikan spasi antara

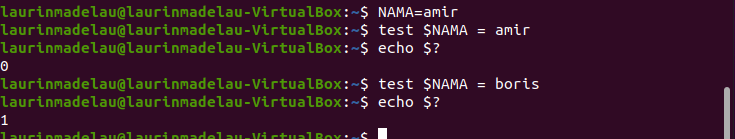
$ NAMA=amir

$ test $NAMA = amir

$ echo $?

$ test $NAMA = boris

$ echo $?



* Gambar di atas merupakan perintah instruksi test. Variabel NAMA diisi dengan amir. Kemudian untuk memanggil variabel NAMA dengan perintah test dengan $NAMA didefiniskan damelia maka status exit bernilai 0. Sedangkan apabila berisi boris status exit berisi 1 artinya nama boris belum diinputkan, maka apabila dipanggil nama boris tidak ditemukan.

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

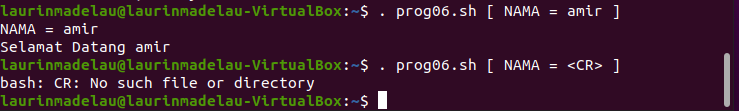


3. Jalankan program prog06.sh dengan memasukkan NAMA = amir

dan NAMA = <CR> perhatikan hasil tampilannya

$ . prog06.sh [NAMA = amir]

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



* Gambar di atas merupakan perintah instruksi test. Perintah ini merupakan eksekusi dari prog06.sh apabila nama diisi dengan amir maka seperti pada perintah 1 yaitu Selamat datang amir, sedangkan pada contoh kedua variabel nama diisi dengan <CR> maka ditampilkan pesan error karena tidak ada di direktori.

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

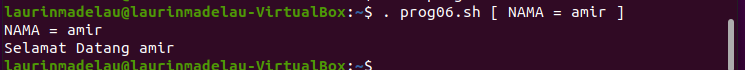
echo “Anda bukan amir, sorry!”

fi



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

$ . prog06.sh [NAMA = amir]



* Gambar di atas merupakan perintah instruksi test. Meskipun perintah if didefiniskan dengan perintah [ ] tapi hasil eksekusi prog06.sh akan menampilkan hasil yang sama dengan perintah test sebelumnya.

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”



* Gambar diatas merupakan perintah notasi && dan ||. 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.

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

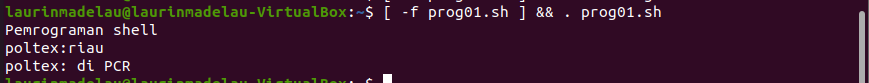
$ [ -f prog99.sh ] && echo “Prog99.sh ada”



* Gambar di atas merupakan perintah notasi && dan ||. File prog99.sh tidak ada, karena itu exit status adalah FALSE dan instruksi echo tidak dijalankan.

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

$ [ -f prog01.sh ] && . prog01.sh



* Gambar di atas merupakan peritnah notasi && dan ||. Pada perintah ini adalah untuk menjalankan shell script dari prog01.sh. sama halnya dengan fungsi instrusi dari . 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 ?”



* Gambar di atas merupakan perintah notasi && dan ||. Bila prog01.sh ada maka jalankan program berikutnya. File prog01.sh memang ada, karena itu exit status adalah TRUE, dan arena sudah TRUE maka instruksi echo tidak lagi dijalankan.

5. File prog99.sh tidak ada, karena itu exit status adalah FALSE, hasil masih tergantung atas exit status instruksi kedua, karena itu instruksi echo dijalankan

$ [ -f prog99.sh ] || echo “Dieksekusi tidak ?”



* Gambar di atas merupakan perintah notasi && dan ||. File prog99.sh tidak ada, karena itu exit status adalah FALSE, hasil masih tergantung atas exit status instruksi ke dua, karena itu instruksi echo dijalankan (dieksekusi).

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

$ [ -f prog99.sh ] || echo “Sorry, prog99.sh tidak ada”



* Gambar di atas merupakan perintah notasi && dan ||. File prog99.sh tidak ada, maka tampilkan pesan error seperti perintah di atas.

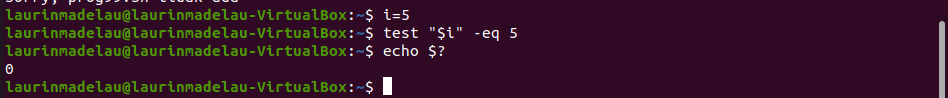
Percobaan 10: Operator bilangan bulat untuk test

1. Menggunakan operator dengan notasi test

$ i=5

$ test “$i” –eq 5

$ echo $?

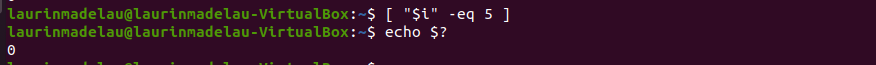


* Gambar di atas merupakan perintah operator bilangan bulat untuk test. Variabel i berisi 5 maka dengan notasi test, variabel i –eq (sama dengan) 5 dan apabila dijalankan instruksi echo $? status exit bernilai 0 karena i memang berisi dengan nilai 5.

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

$ [ “$i” –eq 5 ]

$ echo $?



* Gambar di atas merupakan perintah operator bilangan bulat untuk test. Sama halnya dengan yang nomer 1, namun diganti dengan notasi [ ] maka status exit tetap akan bernilai 0.

3. 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”

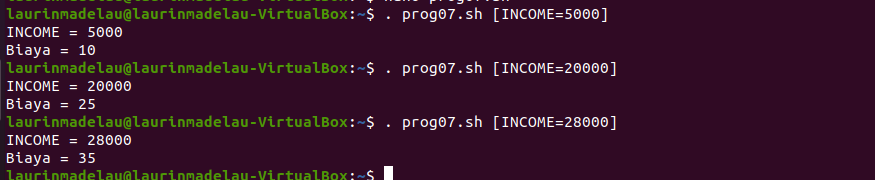


4. Jalankan file prog07.sh dan masukkan untuk INCOME=5000, 20000, 28000

$ . prog07.sh [INCOME=5000]

$ . prog07.sh [INCOME=20000]

$ . prog07.sh [INCOME=28000]



* Gambar di atas merupakan perintah operator logika dan konstruksi elif. Apabila variabel income diisi dengan INCOME antara 0 sampai dengan 10000 maka biaya yang akan ditampilkan adalah 10, sedangkan apabila income berisi nilai antara 10000 sampai dengan 25000 maka biaya yang akan ditampilkan adalah 25 dan jika INCOME berisi lebih dari 25000 maka ditampilkan BIAYA 35 dan tampil seperti gambar di atas.

Percobaan 12: Hitungan aritmatika

1. Menggunakan utilitas expr

$ expr 5 + 1

$ A=5

$ expr $A + 2

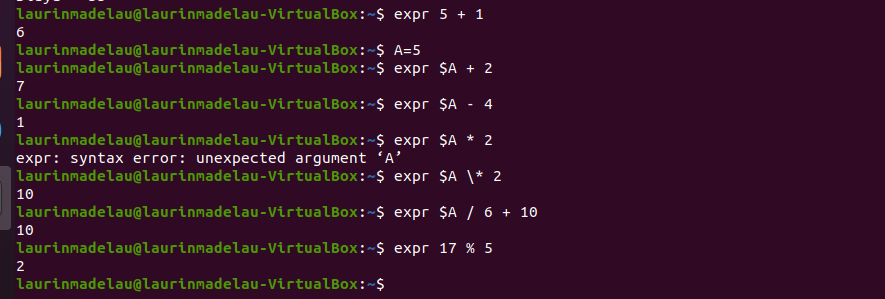
$ expr $A – 4

$ expr $A \* 2 (Ada Pesan Error)

$ expr $A \\* 2

$ expr $A / 6 +10

$ expr 17 % 5



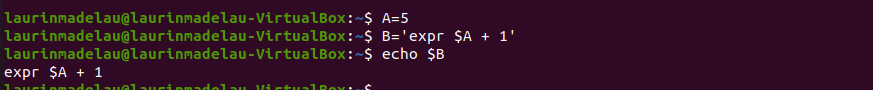
* Gambar di atas merupakan perintah hitungan aritmatika. Untuk perhitungan aritmatika dapat menggunakan utilitas expr. Untuk simbol dari penjumlahan adalah +, pengurangan -, namun untuk perkalian bukan menggunakan perintah \* tetapi expr dari perkalian adalah \\* dan untuk pembagian menggunakan / dan % digunakan untuk expr mod atau sisa hasil bagi dan tampil seperti gambar di atas.

2. Substitusi isi variabel dengan hasil utilitas expr

$ A=5

$ B=`expr $A + 1`

$ echo $B



* Gambar di atas merupakan peritnah hitungan aritmatika. Variabel A diisi dengan 5 kemudian variabel 5 berisi subtitusi dari perintah expr $A + 1 maka apabila variabel B dipanggil akan berisi hasil dari aritmatika tersebut.

Percobaan 13: Instruksi exit

1. Buat shell script prog08.sh

$ vi prog08.sh

#!/bin/sh

if [ -f prog01.sh ]

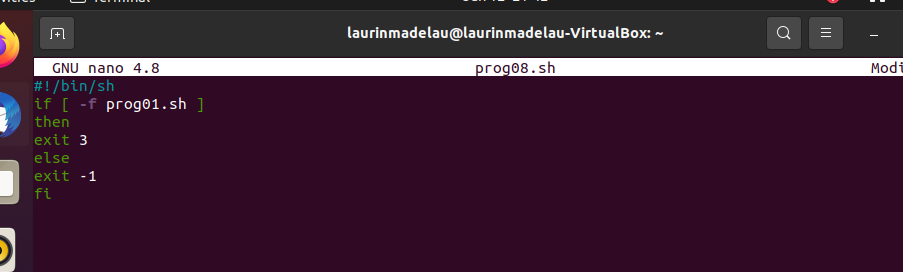
then

exit 3

else

exit –1

fi



2. Jalankan script prog08.sh dan periksa status exit

$ . prog08.sh

$ echo $?





* Gambar di atas merupakan perintah instruksi exit. Apabila prog08.sh dieksekusi dengan . prog08.sh kemudian status exit di tampilkan akan menampilkan nilai 0 dan tampil seperti gambar di atas.

E.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 direktori, beri pesan bahwa file tidak 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

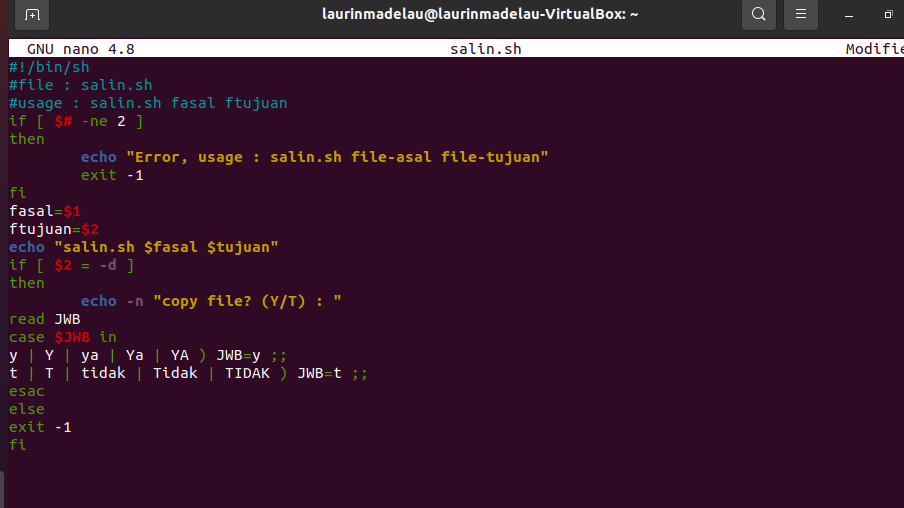
fasal=$1

ftujuan=$2

echo “salin.sh $fasal $ftujuan”

......

......



* Gambar di atas merupakan program copy file, dimana pada gambar di atas jika file tidak berhasil ditemukan maka nantinya akan langsung di close. Untuk pilihan digunakan perintah if else then, tetapi karena perintah pertama pada shell script salin.sh menyatakan jika file tidak ada maka akan di close, dan memang si file ini tidak ada, sehingga perintah yang dikerjakannya pada saat shell script dieksekusi menggunakan perintah . Salin.sh adalah perintah pilihan nomor 1 yaitu “keluar” karena tidak ada filenya dan tampil seperti gambar di atas.

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 logika && atau || pada level shell.

#!/bin/sh

# file: checkdir.sh

# Usage: checkdir.sh DirectoryName

#

if [ $# -ne 1]

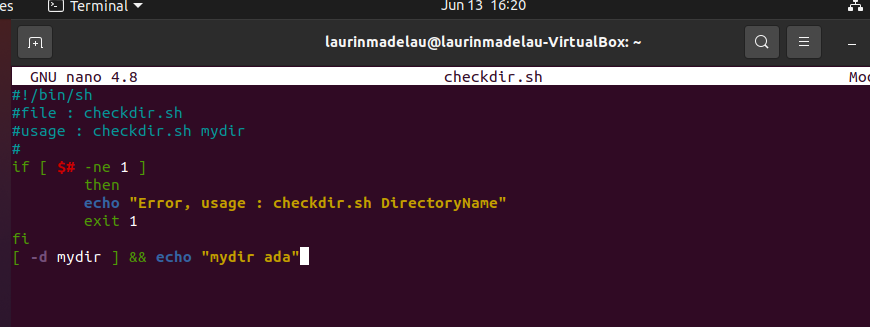
then

echo “Error, usage: checkdir.sh DirectoryName”

exit 1

fi

[ ... ] && ...



* Gambar di atas merupakan program memeriksa nama direktori. Seperti yang terjadi pada latihan no. 1 tadi, ketika dijalankan program checkdir.sh ini tidak menemukan directory yang di cari sehingga secara otomatis akan di “closed” atau exit dan tampil seperti gambar di atas.

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

Solusi:

Variabel: pph = pajak penghasilan

pkp = penghasilan kena pajak

gaji = gaji yan gditerima dalam ribuan rupiah

#!/bin/sh

# prog: pph.sh

echo –n “Berikan gaji dalam ribuan rupiah : “

read gaji

pkp = 10000

if [ $gaji –le $pkp ]

then

pkp=$gaji

pph=`expr $pkp \\* 15 / 100`

else

gaji=`expr $gaji - $pkp`

pph=1500

pkp=25000

if [ $gaji –le $pkp ]

then

pkp=$gaji

pph=`expr $pph + $pkp \\* 25 / 100`

else

pph=`expr $pph + $pkp \\* 25 / 100`

gaji=`expr $gaji - $pkp`

if [ $gaji –gt 0 ]

then

pkp=$gaji

pph=`expr $pph + $pkp \\* 35 / 100`

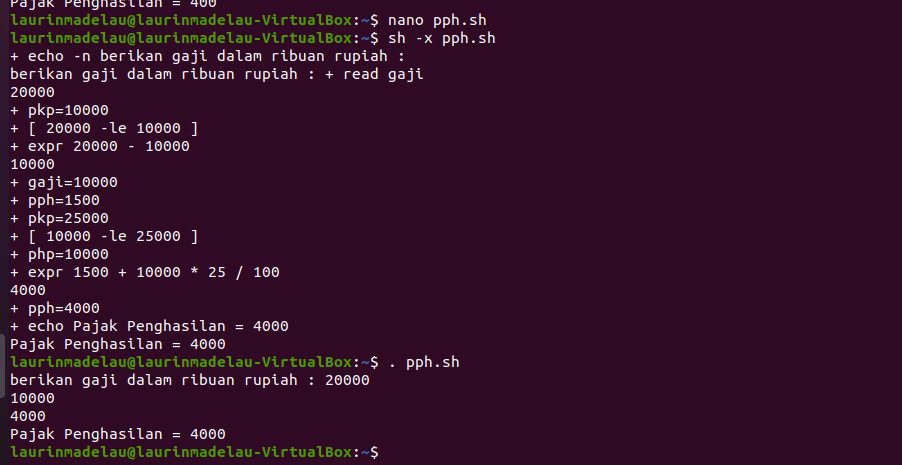
fi

fi

fi

echo “Pajak Penghasilan = $pph”





* Gambar di atas merupakan shell script pph.sh, menghitung PPH per tahun dengan ketentuan 10 juta pertama PPH 15%, 25 juta berikutnya (sisa) PPH 25%, bila masih ada sisa, maka sisa tersebut PPH 35%. Untuk membuat sebuah file pph.sh gunakan perintah vi pph.sh. masuk ke shell dan buat sebuah perintah dalam vi pph.sh dengan memasukan perintah yang ada pada gambar di atas. Gunaka tombol Esc ‘:’ dan wq untuk keluar dari shell dan menyimpan. Kemudian jalankan pph.sh dengan \ perintah bash pph.sh dan untuk debugging untuk melakukan tracing (debug) gunakan opsi –x pada eksekusi shell, maka hasilnya tampil seperti gambar di atas.