


MODUL PRAKTIKUM

“SISTEM OPERASI”



**LABORATORIUM KOMPUTER
FAKULTAS ILMU KOMPUTER
UNIVERSITAS SRIWIJAYA
2020**

 Universitas Sriwijaya Fakultas Ilmu Komputer Laboratorium	LEMBAR PENGESAHAN MODUL PRAKTIKUM		SISTEM MANAJEMEN MUTU ISO 9001:2008
No. Dokumen	Tanggal	2 JUNI 2013
Revisi	0	Halaman	ii DARI 124

MODUL PRAKTIKUM

Mata Kuliah Praktikum : Sistem Operasi
Kode Mata Kuliah Praktikum : FTK07411
SKS : 2
Program Studi : Sistem Komputer
Semester : 3 (Ganjil)

DIBUAT OLEH	DISAHKAN OLEH	DIKETAHUI OLEH
TIM LABORAN LABORATORIUM FASILKOM UNSRI	TIM DOSEN TEKNIK KOMPUTER FASILKOM UNSRI	KEPALA LABORATORIUM

Daftar Isi

Cover	i
Lembar Pengesahan	ii
Daftar Isi	iii
Praktikum 1	1
Praktikum 2	14
Praktikum 3	23
Praktikum 4	41
Praktikum 5	51
Praktikum 6	64
Praktikum 7	72
Praktikum 8	84
Praktikum 9	109

Praktikum 1 Instalasi Sistem Operasi Linux

A. TUJUAN

- Mengetahui prosedur instalasi pada sistem operasi linux
- Mampu menjalankan instalasi melalui *Graphic User Interface* (GUI) maupun *Command Line* Linux
- Mampu menganalisis proses instalasi

B. DASAR TEORI

Sistem operasi adalah seperangkat program terstruktur yang mengelola sumber daya perangkat keras (*Hardware*) dan menyediakan layanan umum untuk aplikasi perangkat lunak (*Software*) dan sistem operasi merupakan hal yang paling penting dari perangkat lunak dalam system computer. Sistem operasi mempunyai penjadwalan yang sistematis mencakup perhitungan penggunaan memori, pemrosesan data, penyimpanan data dan sumber daya lainnya.

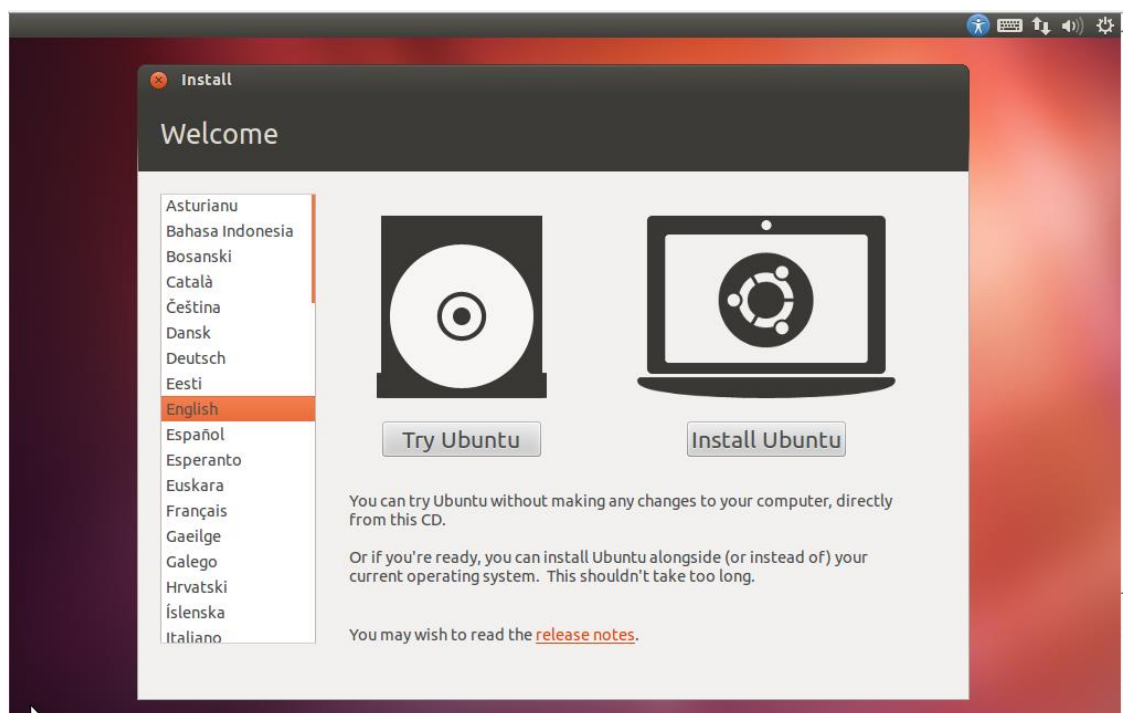
Linux merupakan sistem operasi yang dapat dikendalikan oleh salah satu atau lebih antarmuka baris perintah (*Command Line Interface* atau *CLI*) berbasis teks, antarmuka pengguna grafis (*Graphics User Interface* atau *GUI*, yang umumnya merupakan konfigurasi bawaan untuk versi desktop). Inti dari linux sendiri adalah *KERNEL* (bagian inti dari sistem operasi). Salah satu system operasi linux yaitu Ubuntu.

C. LANGKAH – LANGKAH

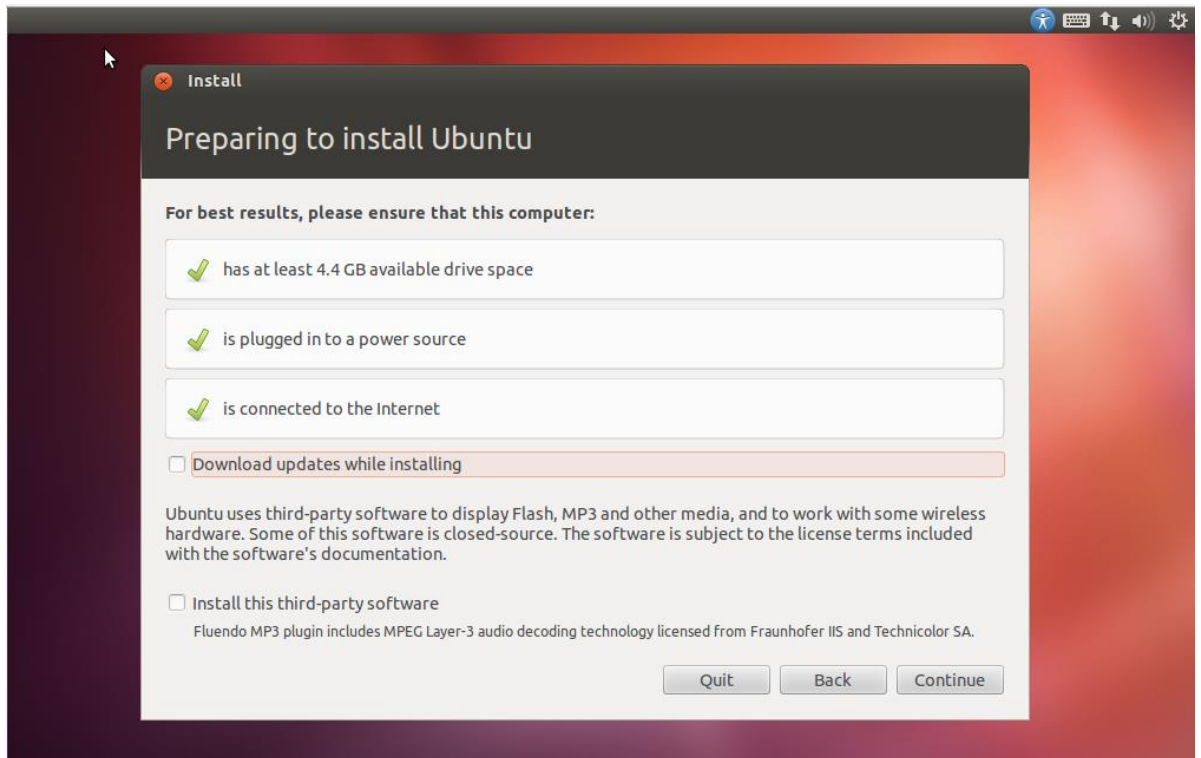
1. Masukkan CD Ubuntu 12.04 installer yang sudah di *burn* dalam CD, sebelumnya atur di bios untuk pengaturan *booting* dengan CD-ROOM sebagai primer pertama booting.
2. Tunggu proses *booting* sampai keluar tampilan



3. Pilih bahasa dan pilih opsi **Install Ubuntu** untuk memulai proses instalasi, namun jika ingin mencoba tanpa melakukan instalasi pilih **Try Ubuntu**



4. Pilihan berikutnya yaitu memilih opsi paket pendukung seperti plugins mp3 melalui jaringan internet, namun apabila tidak terkoneksi dalam internet tidak perlu memberi tanda, kemudian pilih **Continue**



5. Tahap berikutnya adalah proses pemilihan partisi harddisk instalasi dengan pilihan,

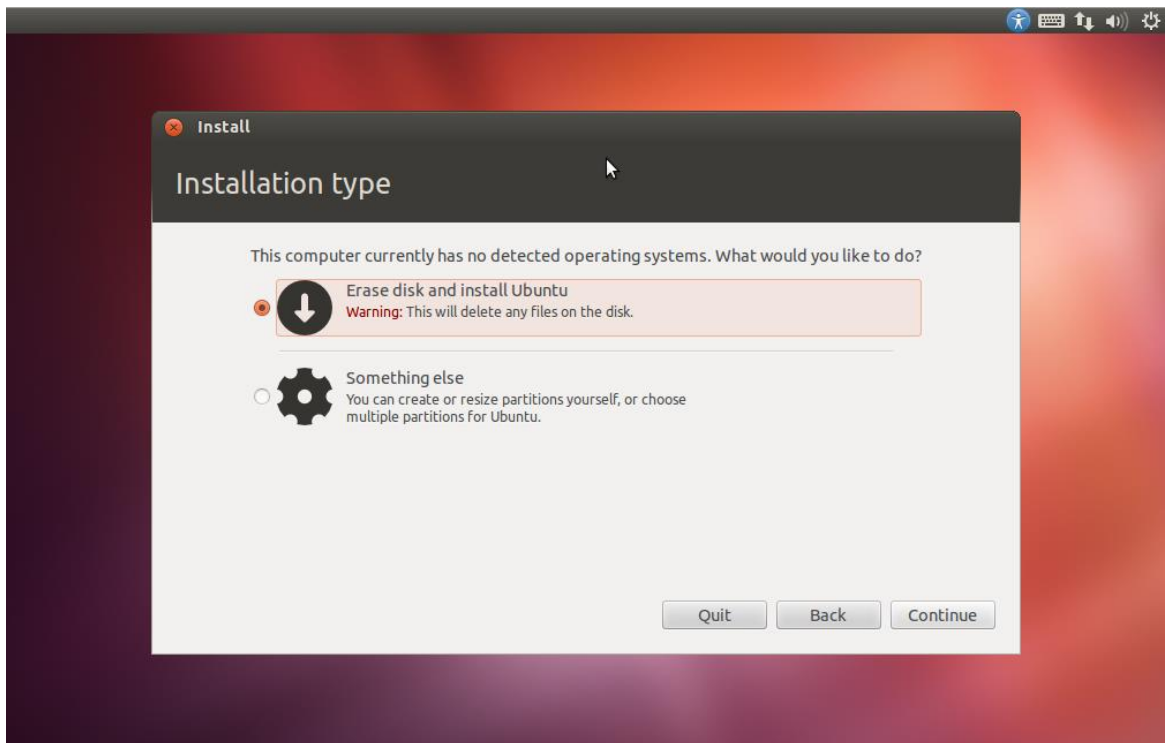
- ***Erase disk and install Ubuntu***

Untuk Pilihan tersebut merupakan menginstal sistem operasi Linux Ubuntu menggunakan partisi harddisk sepenuhnya tanpa ada pemilihan secara manual.

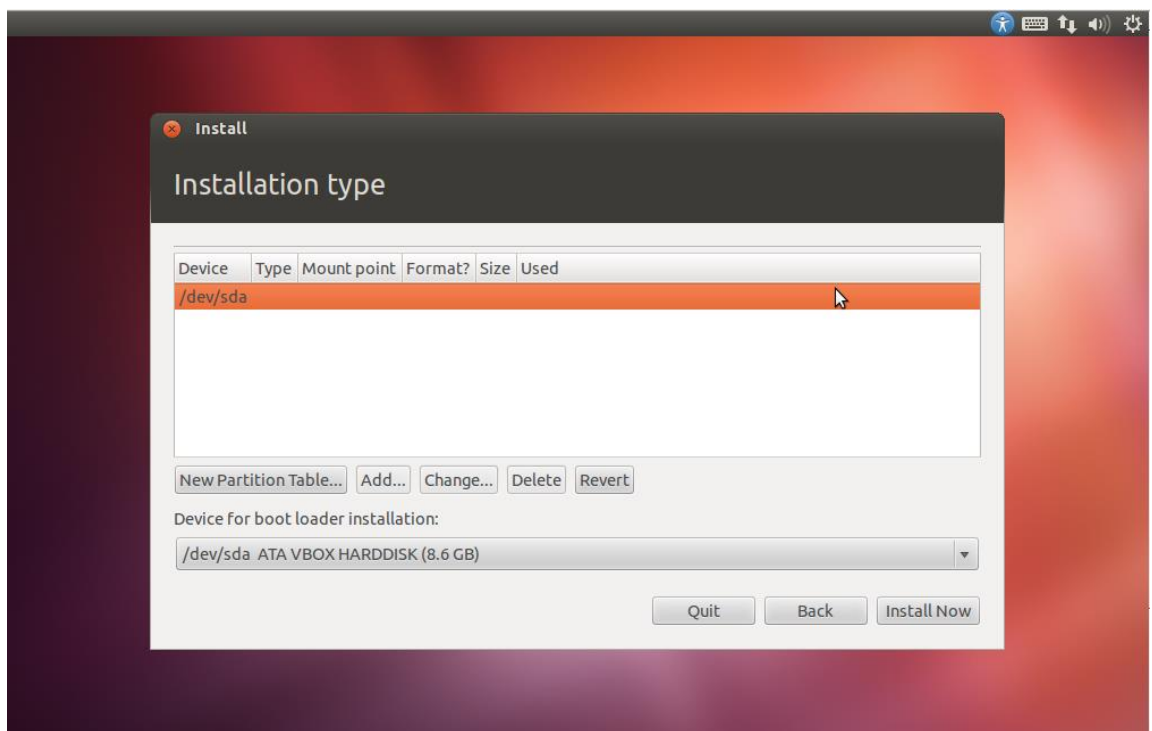
- ***Something else***

Merupakan pilihan menginstalasi sistem operasi Linux Ubuntu secara manual untuk pemilihan partisi harddisk

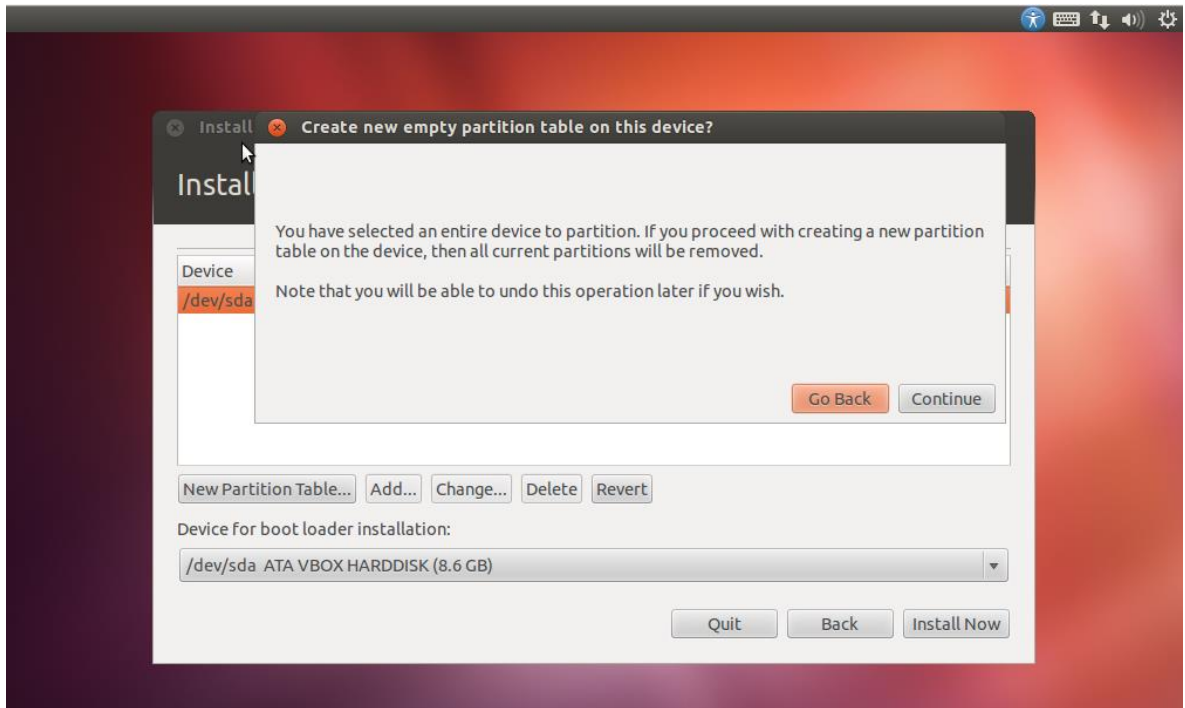
Pilih opsi **Something else** untuk instalasi yang akan dilakukan, sehingga bisa manage partisi harddisk untuk keperluan tempat filesystem dan penyimpanan data. Klik **Continue**



6. Tampilan berikutnya merupakan manage partisi harddisk untuk dilakukannya instalasi sistem operasi.

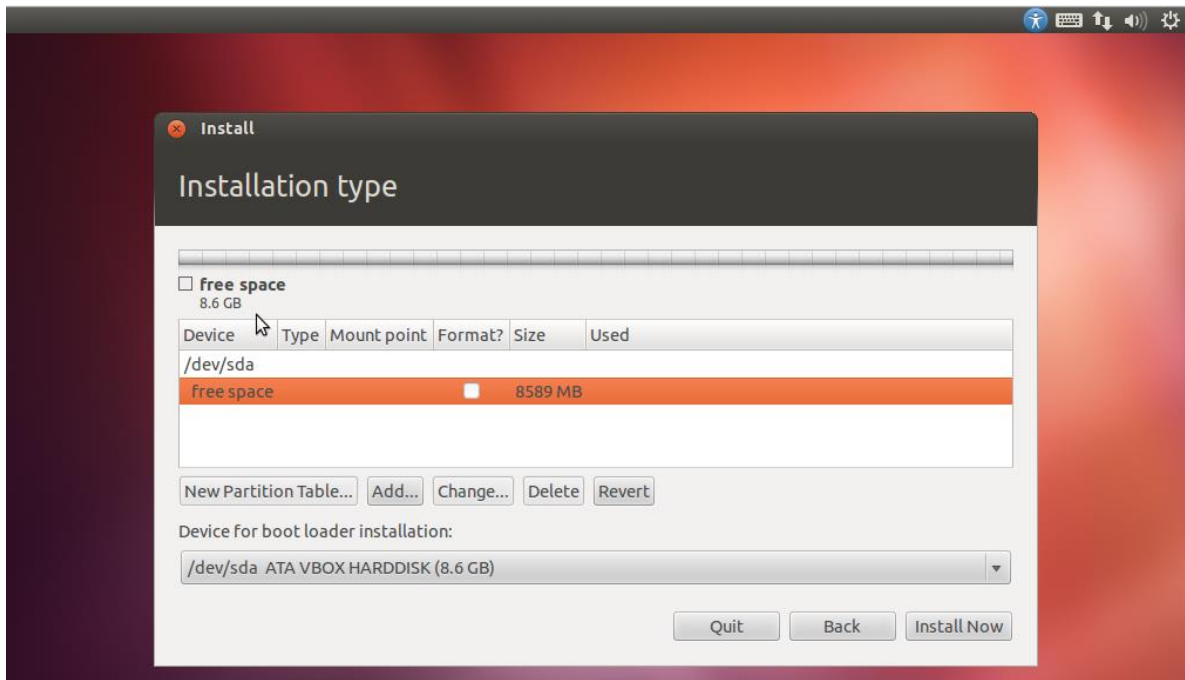


7. Klik **New Partition Table**

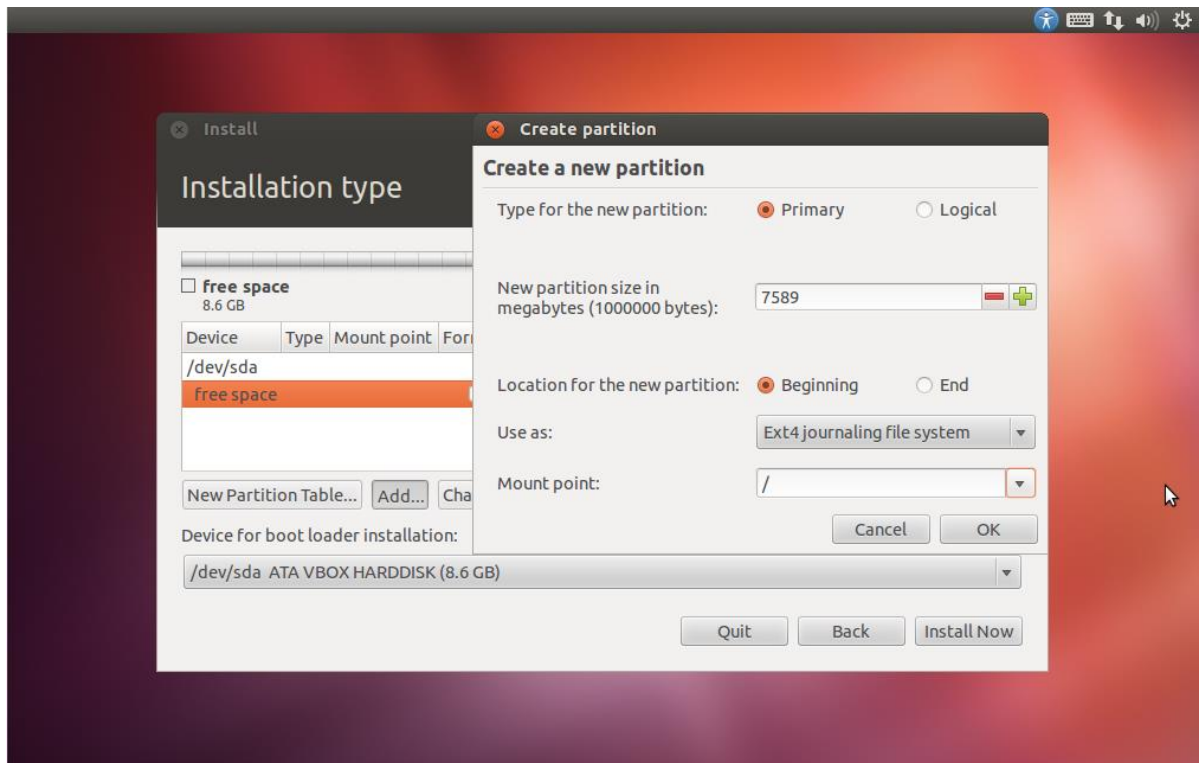


Kemudian klik **Continue**

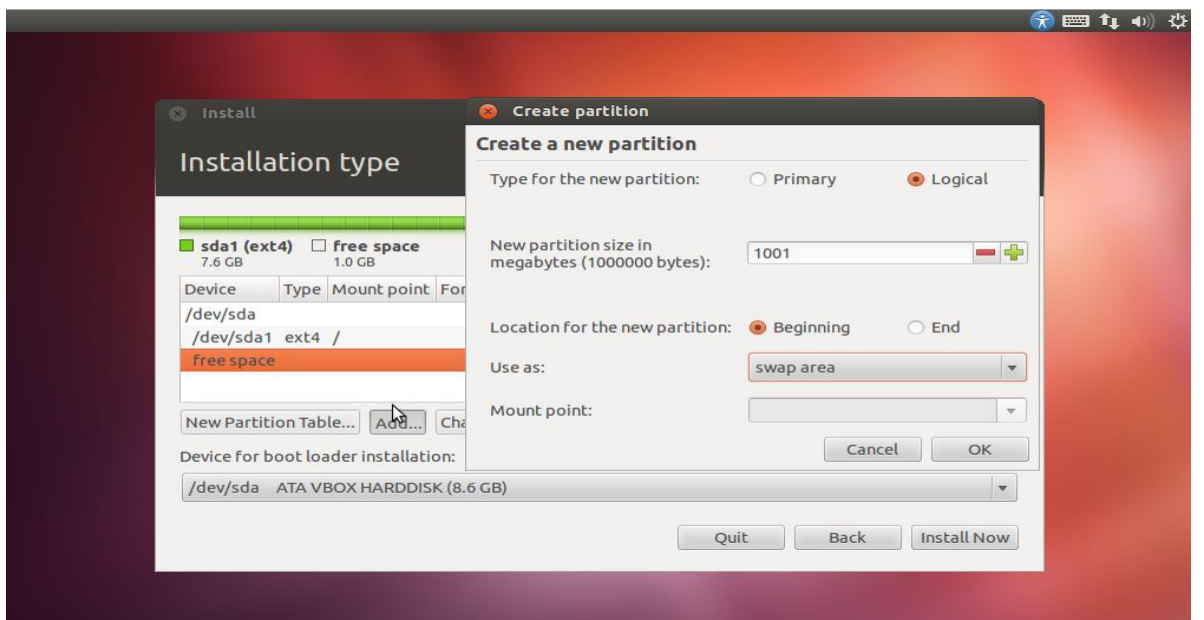
8. Maka tampilan berikutnya akan menampilkan partisi yang sudah terbuat (FreeSpace)



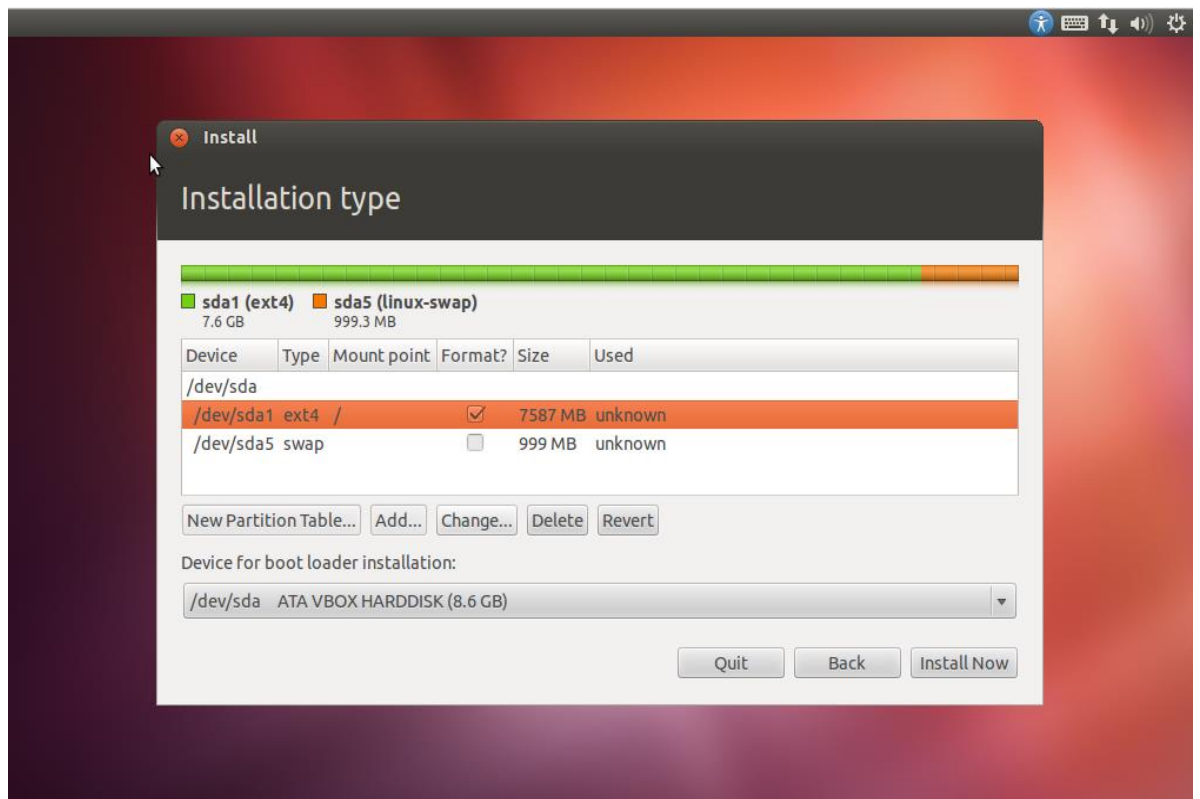
9. Selanjutnya pilih **Add** dan untuk file system berikan *space* harddisk lebih besar misalkan 8 GB ataupun lebih. Kemudian pada opsi **Use As** pilih dengan format **Ext4 Journal System** dan opsi **Mount Point** pilih **"/"** (root). Selesai **OK**



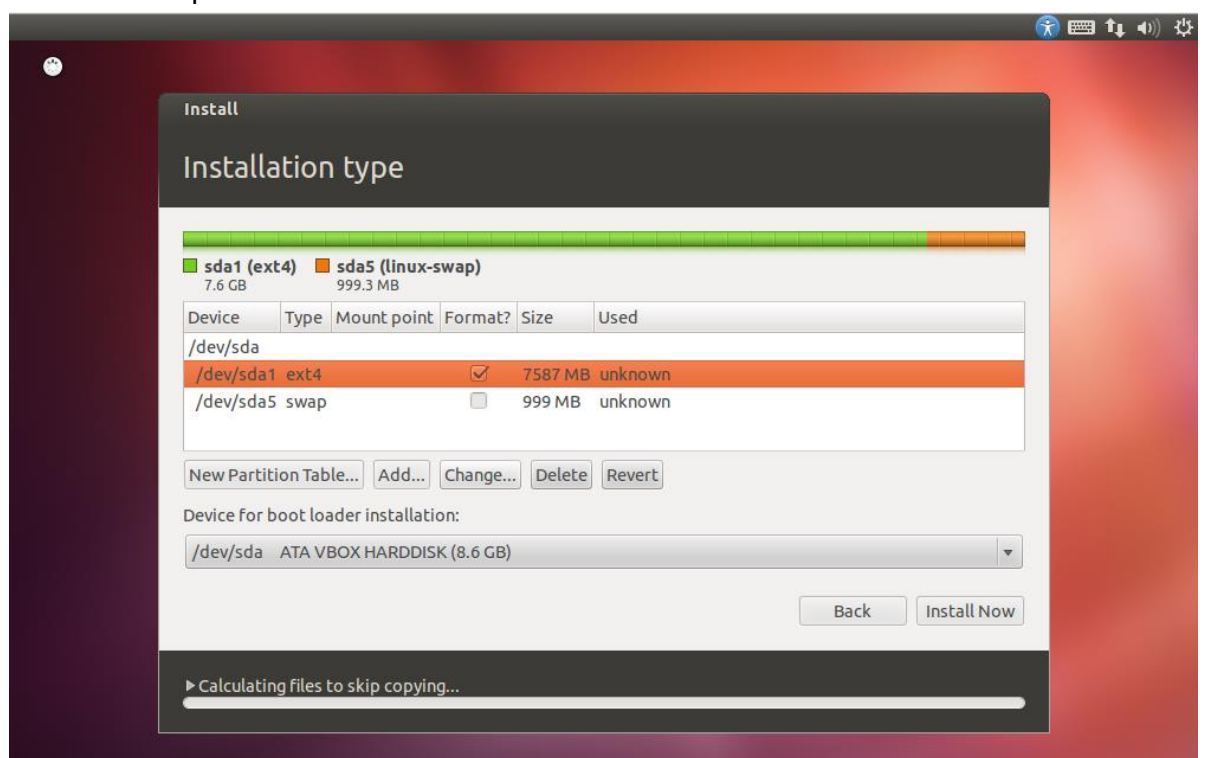
10. Setelah selesai sisa partisi harddisk dibuat untuk partisi **swap** dengan catatan untuk partisi ukuran partisinya lebih kecil dari ukuran harddisk filesystem. Pada **Use As** pilih **Swap Area**. Selesai Klik **OK**



11. Berikutnya Partisi harddisk siap dilakukan instalasi sistem operasi Linux Ubuntu
12.04. Klik **Install Now**



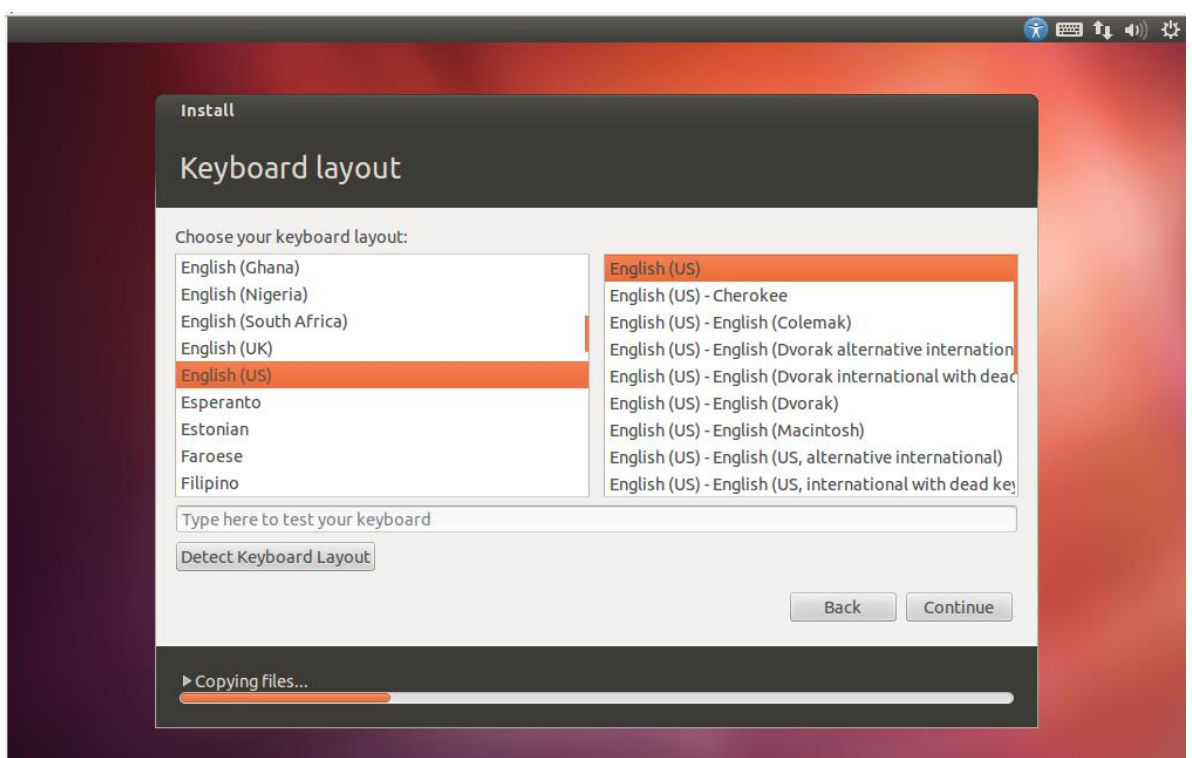
12. Setelah itu proses Instalasi ke harddisk akan dilakukan.



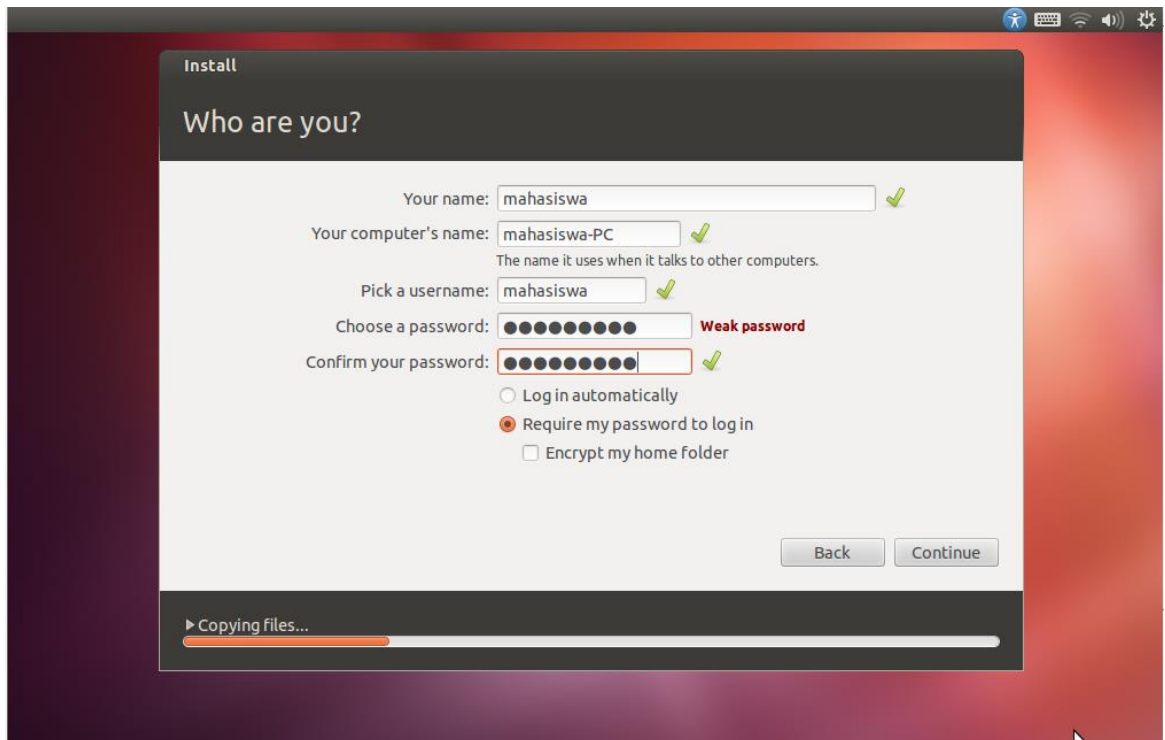
13. Saat melakukan proses instalasi, setting untuk zona waktu sesuai yang digunakan. Klik **Continue**



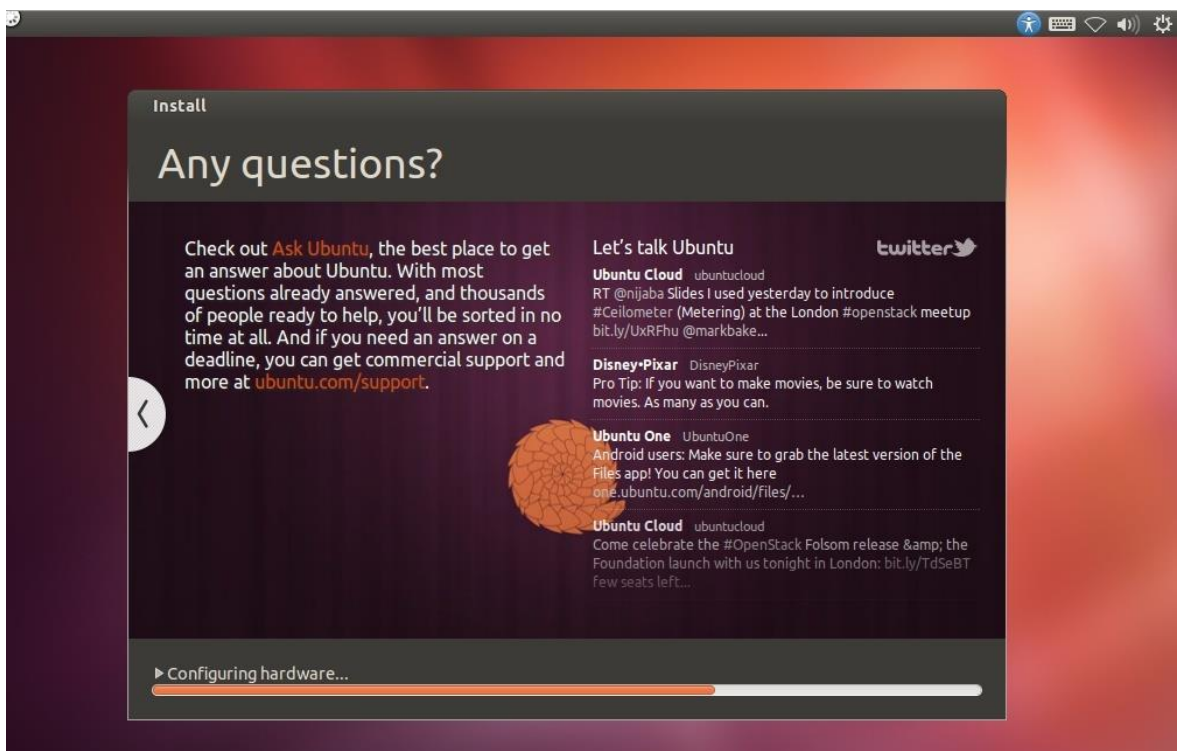
14. Kemudian masih dalam proses instalasi, ditampilkan pilihan tipe *keyboard* yang di gunakan. Selesai, klik **Continue**



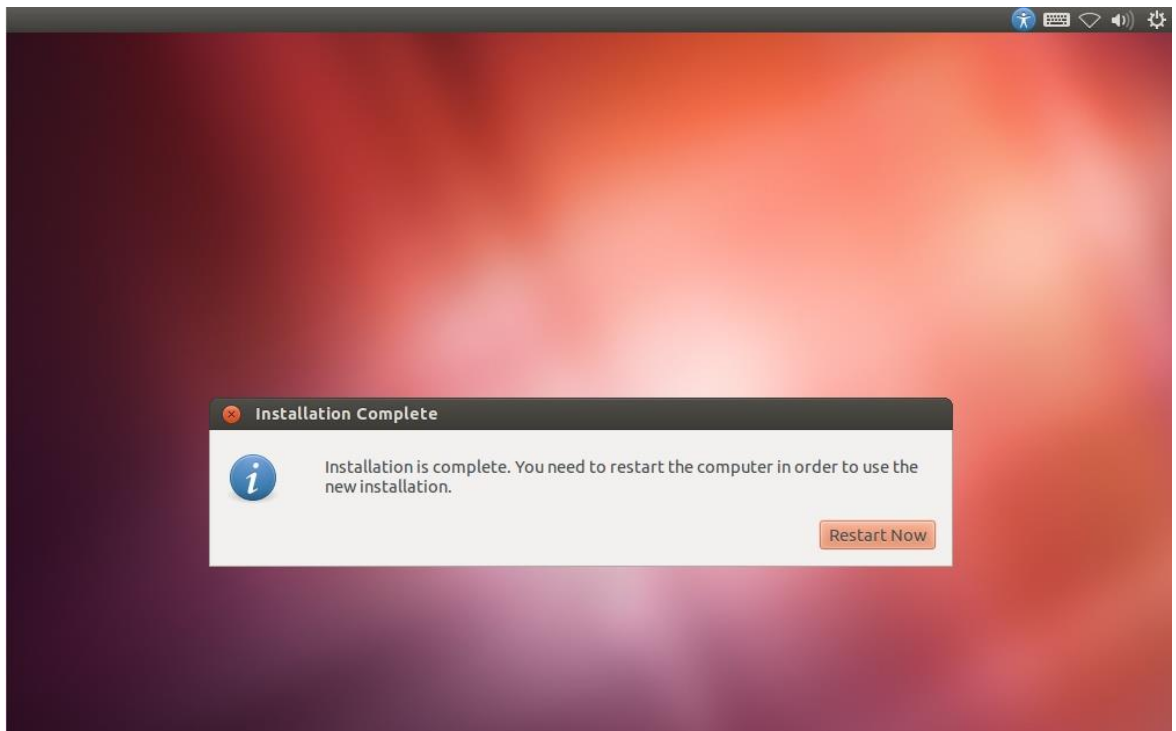
15. Tampilan berikutnya merupakan input **username** dan **password** untuk verifikasi saat masuk atau *login* ke sistem operasi. Apabila telah selesai klik **Continue**



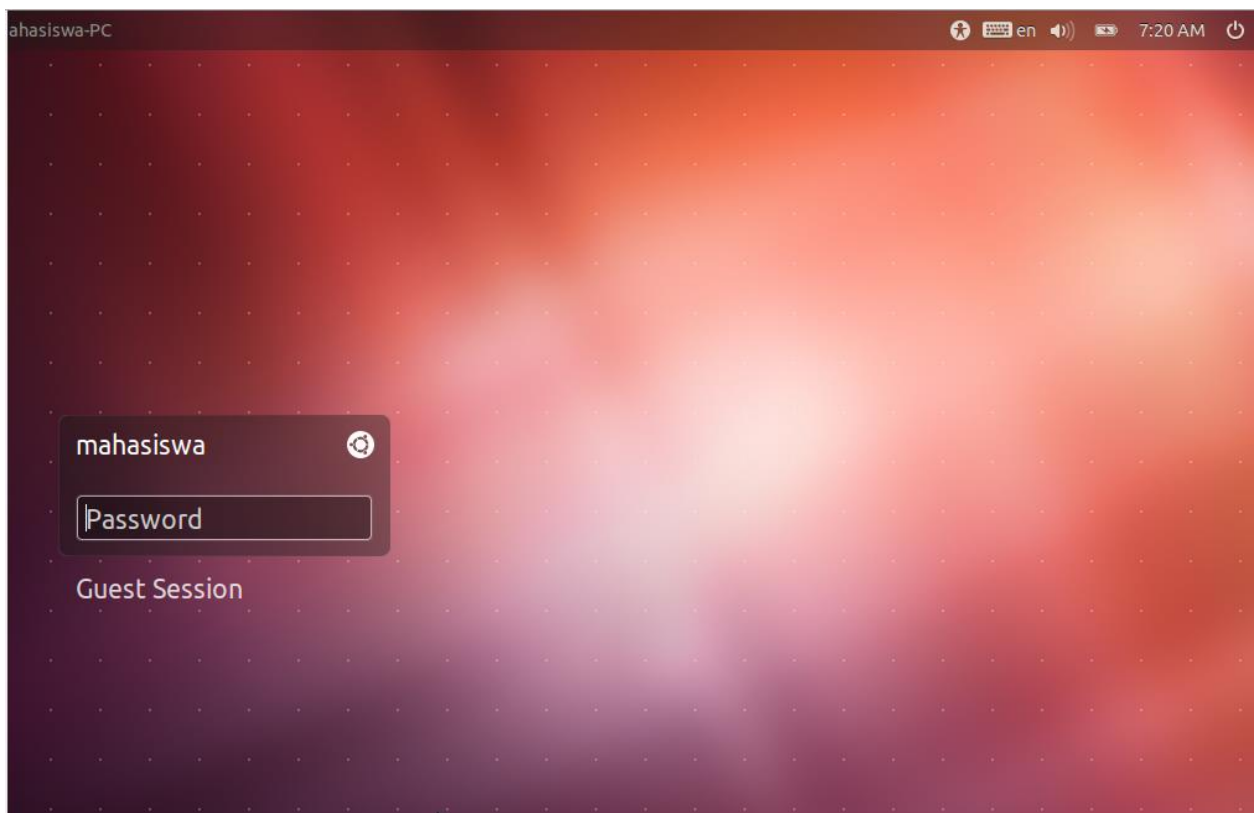
16. Setelah itu akan memasuki proses peng-copy-an file dan instalasi ke *filesystem*.



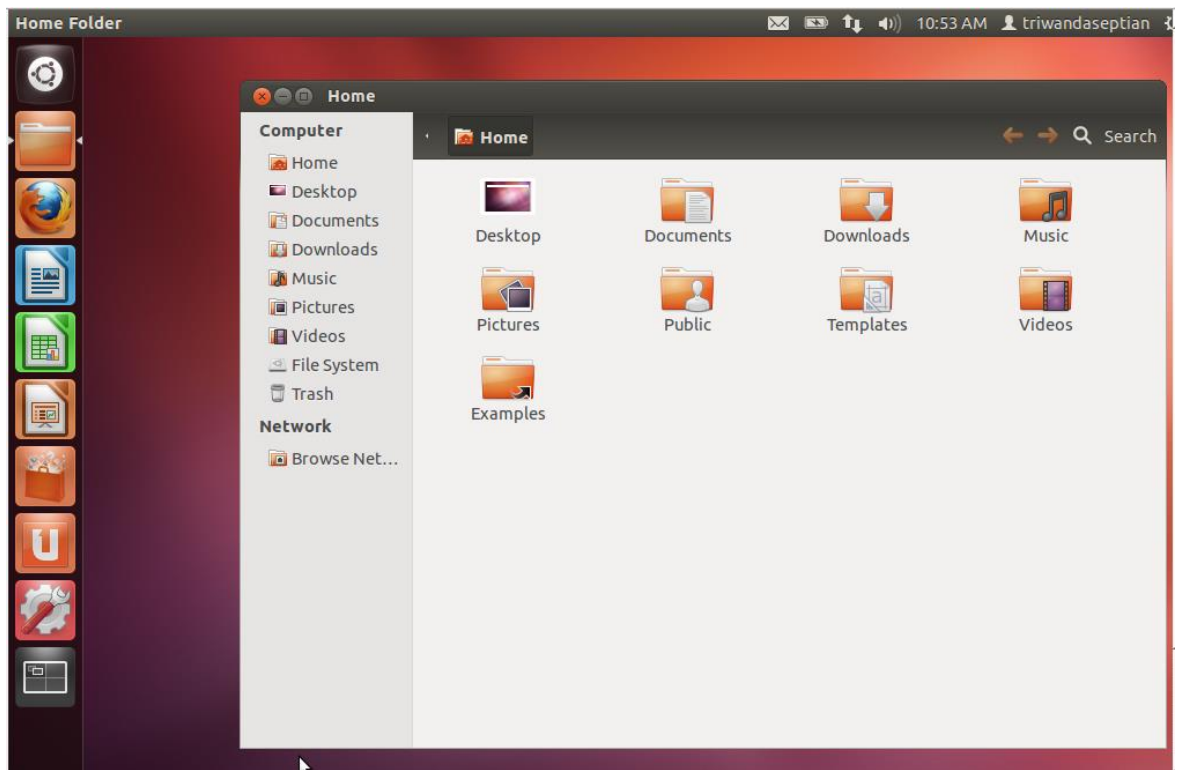
17. Setelah proses *install*, maka *restart* komputer dengan memilih **Restart Now**.



18. Masuk ke sistem operasi yang sudah terinstall dan masukkan **username** dan **password** yang telah dibuat.



19. Tampilan sistem operasi Ubuntu 12.04 desktop yang telah terinstall

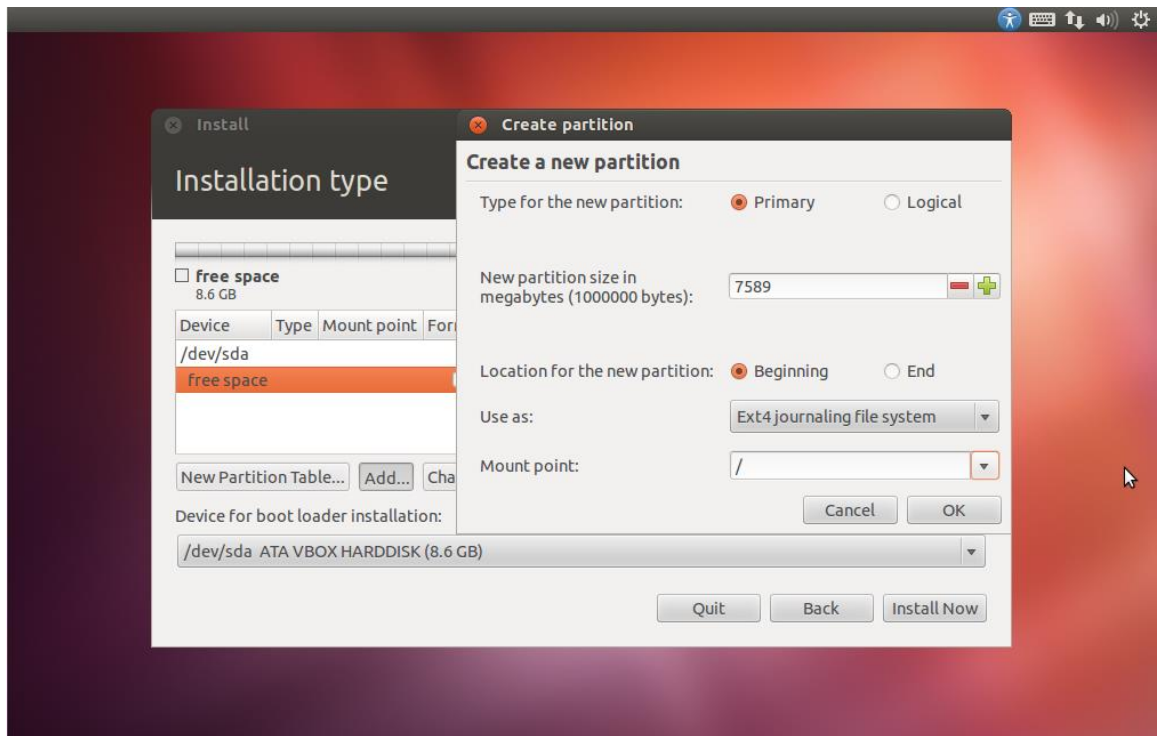


D. PERCOBAAN

1. Lakukan proses Instalasi sesuai dengan langkah – langkah !

E. TUGAS

1. Analisislah pada gambar kenapa saat instalasi perlu dipilih “/” pada opsi **Mount Point** ?



2. Berikan penjelasan tentang **ext4**, **ext3**, **swap**, **ntfs**, **fat32**, **btrfs** !

F. TUJUAN

- Mengetahui format intruksi arsitektur sistem pada sistem operasi Linux
- Mempelajari Utilitas dasar pada sistem operasi Linux
- Menggunakan perintah – perintah dasar pada sistem operasi Linux

G. DASAR TEORI

Linux yang pada dasarnya untuk menjalankan setiap service dengan menjalankan *Command line* atau baris perintah dalam lingkungan *shell*. Keuntungan menggunakan perintah di baris perintah adalah efektifitas dan maksimalitas kerja. Prompt dari shell bash pada linux menggunakan “\$”.

```
username@linux-PC:~$
```

Untuk sebuah sesi linux terdiri dari :

1. Login
2. Bekerja dengan shell atau menjalankan aplikasi
3. logout

Seperti halnya mengetik perintah di DOS, baris perintah di linux juga diketik di prompt yang ada di lingkungan *shell* dan diakhiri dengan enter untuk mengeksekusi perintah tersebut.

I. Perintah – Perintah Dasar

ls	Melihat isi direktori
mkdir	Menciptakan direktori
cd	Membuka direktori
rmdir	Menghapus direktori
Cat	Menampilkan isi file
cp	Menyalin (copy) file
mv	Mengganti nama file/direktori dan memindahkan file ke direktori lain
ln	Link ke file lain
lp	Mencetak isi file
find	Mencari file
chmod	Untuk mengubah model akses terhadap file atau direktori

chgrp	Mengubah group file
touch	Membuat file

II. Format Intruksi Linux

Intruksi linux standar mempunyai format sebagai berikut :

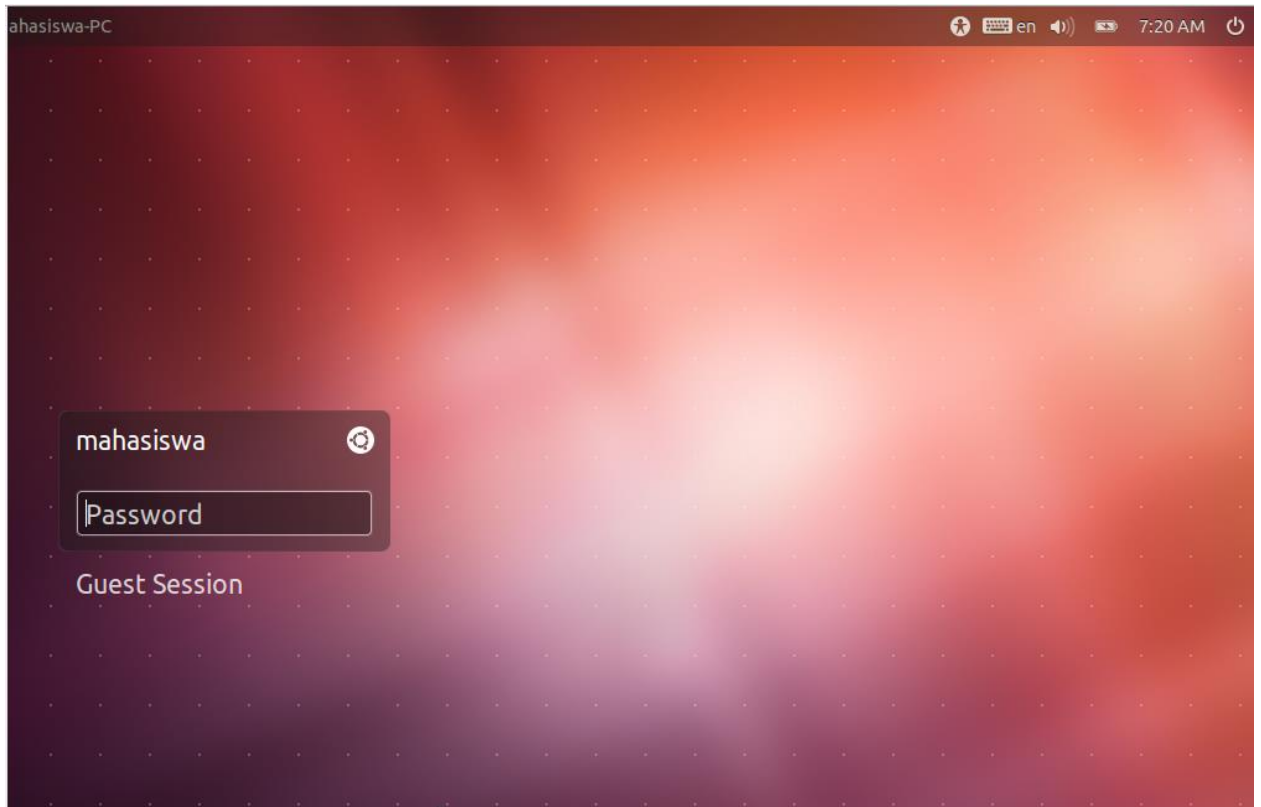
```
$ NamaIntruksi [Pilihan] [argument]
```

Pilihan adalah opsi yang dimulai dengan tanda – (minus). Argumen dapat kosong, atau beberapa argument (parameter). Contoh :

\$ ls	tanpa argumen
\$ ls -a	option adalah -a = all, tanpa argumen
\$ ls /bin	tanpa option, argumen adalah /bin
\$ ls /bin/etc/usr	ada 3 argumen
\$ ls -l /usr	1 option dan 1 argumen l = long list
\$ ls -la /bin/etc	2 option -l dan -a dan 2 argumen
\$ rm myfile	menghapus file myfile
\$ rm -rf mydir	menghapus direktori mydir dan semua file didalamnya
\$ cat myfile	menampilkan isi file myfile

H. LANGKAH – LANGKAH

1. Hidupkan komputer
2. Masuk ke sistem operasi linux
3. Tunggu sampai ada perintah login untuk mengisi nama user dan perintah password untuk mengisi password dari user.
 - Tampilan login dengan tampilan GUI



- Tampilan login berupa command line tanpa GUI

```
Ubuntu 12.04 LTS mahasiswa-PC tty1
mahasiswa-PC login: mahasiswa
Password: _
```

4. Untuk keluar dari system gunakan perintah Logout atau Exit
5. Gunakan perintah – perintah untuk informasi user :
Id, hostname, uname, w, who, whoami, chfn.
6. Gunakan perintah – perintah dasar (basic command):
date, cal, man, clear, apropos, whatis
7. Gunakan perintah – perintah dasar untuk manipulasi file :
ls, file, cat, more, pg, cp, mv, rm, grep

I. PERCOBAAN

1. Percobaan 1 : Meilhat identitas diri (nomor id dan group id)
\$ id

2. Percobaan 2 : Melihat tanggal dan kalender dari system

- Melihat tanggal saat ini

```
$ date
```

- Melihat kalender

```
$ cal 10 2012
```

```
$ cal -y
```

3. Percobaan 3 : Melihat identitas mesin

```
$ hostname
```

```
$ uname
```

```
$ uname -a
```

4. Percobaan 4 : Melihat siapa yang sedang aktif

1. Mengetahui siapa saja yang sedang aktif

```
$ w
```

```
$ who
```

```
$ whoami
```

2. Mengubah informasi finger

```
$ chfn mahasiswa
```

```
Changing finger information for student.
```

```
Password:
```

```
Name[Student]: <Nama Pengguna>
```

```
Office[ ]: Lab Linux
```

```
Office Phone [ ]: 9999999
```

```
Home Phone [ ]: 8888888
```

```
Finger information changed
```

5. Percobaan 5 : Menggunakan Manual

```
$ man ls
```

```
$ man man
```

```
$ man -k file
```

```
$ man 5 passwd
```

6. Percobaan 6 : Menghapus layar

```
$ clear
```

7. Percobaan 7 : Mencari perintah yang deskripsinya mengandung kata kunci yang dicari.

```
$ apropos date
```

```
$ apropos mail
```

```
$ apropos telnet
```

8. Percobaan 8 : Mencari perintah yang tepat sama dengan kunci yang dicari.

```
$ whatis date
```

9. Percobaan 10 : Manipulasi berkas (file) dan direktori

1. Menampilkan curen working directory

```
$ ls
```

2. Melihat semua file lengkap

```
$ ls -l
```

3. Menampilkan semua file atau direktori yang tersembunyi

```
$ ls -a
```

4. Menampilkan semua file atau direktori tanpa proses sorting

```
$ ls -f
```

5. Menampilkan isi suatu direktori

```
$ ls /usr
```

6. Menampilkan isi direktori root

```
$ ls /
```

7. Menampilkan semua file atau direktori dengan menandai : tanda (/) untuk direktori, tanda asterik (*) untuk file yang bersifat executable, tanda (@) untuk file symbolic link, tanda (=) untuk socket, tanda (%) untuk whiteout dan tanda (|) untuk FIFO.

```
$ ls -F /etc
```

8. Menampilkan file atau direktori secara lengkap yaitu terdiri dari nama file, ukuran, tanggal dimodifikasi, pemilik, group dan mode atau atributnya.

```
$ ls -l /etc
```

9. Menampilkan semua file dan isi direktori. Argumen ini akan menyebabkan proses berjalan agak lama, apabila proses akan dihentikan dapat menggunakan ^c

```
$ ls -R /usr
```

10. Percobaan 11 : Melihat tipe file

```
$ file
```

```
$ file *
```

```
$ file /bin/ls
```

11. Percobaan 12 : Menyalin file

1. Mengkopi suatu file. Berikan opsi -i untuk pertanyaan interaktif bila file sudah ada.

```
$ cp /etc/group f1
```

```
$ ls -l
```

```
$ cp -i f1 f2
```

```
$ cp -i f1 f2
```

2. Mengkopi ke direktori

```
$ mkdir backup
```

```
$ cp f1 f3
```

```
$ cp f1 f2 f3 backup
```

```
$ ls backup
```

```
$ cd backup
```

```
$ ls
```

12. Percobaan 13 : Melihat isi file

1. Menggunakan instruksi cat

```
$ cat f1
```

2. Menampilkan file per satu layar penuh

```
$ more f1
```

13. Percobaan 14 : Mengubah nama file

1. Menggunakan instruksi mv

```
$ mv f1 prog.txt
```

```
$ ls
```

2. Memindahkan file ke direktori lain. Bila argumen terakhir adalah nama direktori, maka berkas-berkas akan dipindahkan ke direktori tersebut.

```
$ mkdir mydir
```

```
$ mv f1 f2 f3 mydir
```

14. Percobaan 15 : Menghapus file

```
$ rm f1
```

```
$ cp mydir/f1 f1
```

```
$ cp mydir/f2 f2
```

```
$ rm f1
```

```
$ rm -i f2
```

15. Percobaan 16 : Mencari kata/kalimat dalam file

```
$ grep root /etc/passwd
```

```
$ grep ":0:" /etc/passwd
```

```
$ grep mahasiswa /etc/passwd
```

E. TUGAS

1. Tugas Percobaan 1 Informasi finger

Ubahlah informasi **finger** pada komputer Anda.

2. Tugas Percobaan 2 log user aktif

Lihatlah user-user yang sedang aktif pada komputer Anda.

3. Tugas Percobaan 3 group

Buka file `$cat /etc/group` kemudian analisa untuk `root:x:0`

LAPORAN HASIL PRAKTIKUM

Nama :

Nim :

Judul Percobaan :

Hasil Percobaan :

Analisis Percobaan :

Kesimpulan Percobaan :

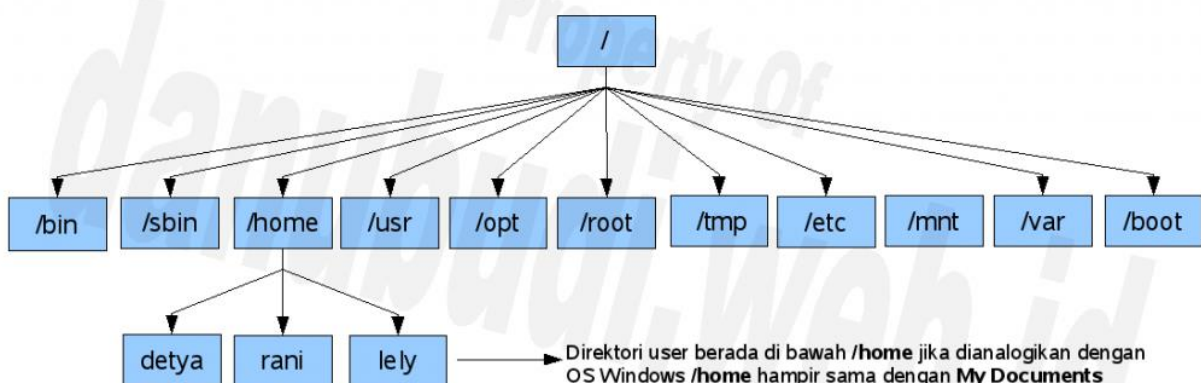
J. TUJUAN

- Mengenal organisasi File di Linux
- Menciptakan dan manipulasi direktori
- Mempelajari ijin akses (permission) dari file dan direktori
- Mengenal konsep Owner dan Group
- Mengerti konsep Link dan symbolic link

K. DASAR TEORI

Sistem file pada Linux menyerupai pepohonan (tree), yaitu dimulai dari root, kemudian direktori dan sub direktori. Sistem file pada Linux diatur secara hirarkhikal, yaitu dimulai dari root dengan symbol “/”.

STRUKTUR DIREKTORI LINUX SECARA UMUM



nb : walaupun ketiga user diatas berada di dalam direktori yang sama (/home) namun tiap user memiliki direktori terpisah sesuai dengan nama user dan tiap user tidak bisa melihat isi dari direktori user lainnya

Berkas-berkas atau file dapat disimpan pada direktori root, tetapi usahakan tidak menyimpan berkas-berkas biasa sehingga direktori ini tetap terjaga keteraturannya.

Perubahan penamaan direktori-direktori yang ada pada direktori root akan menyebabkan sebagian besar dari sistem menjadi tidak berguna. Karena sebagian besar dari direktori-direktori ini berisi fungsi-fungsi yang sifatnya kritis, dimana sistem operasi dan

semua aplikasi memerlukan direktori-direktori ini dengan nama yang sudah diberikan pada awal instalasi. Tetapi kita bisa membuat direktori lain pada level ini. Direktori home juga bisa ditemukan pada level ini hasil pembuatan oleh administrator sistem.

DIREKTORI STANDAR

***/** : menunjukkan hirarki tertinggi dari sistem direktori Linux dimana direktori ini membawahi dari direktori `/usr`, `/home`, `/mnt` dan direktori lainnya seperti gambar diatas.

***/bin** : berisi program yang berisi perintah-perintah yang digunakan oleh user biasa seperti perintah `ls` (menampilkan isi dari suatu direktori, `cd` (untuk berpindah direktori). Intinya direktori ini berisi program-program yang esensial agar sistem operasi dapat bekerja dengan benar. Dalam direktori ini dapat ditemukan perintah-perintah navigasi, program-program shell, perintah pencarian. Bin adalah singkatan dari kata binary.

Di Linux sebuah binary adalah file yang dapat dieksekusi (executable). Sebagian besar dari perintah dalam UNIX merupakan binary, perintah-perintah tersebut merupakan program-program kecil yang dapat dieksekusi oleh pengguna. Ada beberapa perintah yang disebut perintah built-in dimana fungsi tersebut dikendalikan oleh program shell, sehingga mereka tidak beroperasi sebagai binary yang terpisah. Terkadang direktori bin terhubung ke direktori lain yang dinamakan `/usr/bin`. Direktori `/usr/bin` biasanya adalah lokasi sebenarnya dari binary-binary pengguna disimpan. Dalam hal ini, `/bin` adalah gerbang untuk mencapai `/usr/bin`.

Contoh command yang ada di direktori `/bin`. `login`, Shell (`bash`, `ksh`, `csh`), File utility (`cp`, `mv`, `rm`, `ln`, `tar`), Editor (`ed`, `vi`), Filesystem utility (`dd`, `df`, `mount`, `umount`, `sync`), System utility (`uname`, `hostname`, `arch`), GNU utility (`gzip`, `gunzip`).

***/sbin** : berisi program yang berisi perintah-perintah yang digunakan oleh super user seperti `ifconfig` (menampilkan informasi tentang kartu jaringan / network device yang terpasang pada mesin). Contoh program yang di jalan dari direktori `/sbin`, antara lain : `fsck`, `fdisk`, `mkfs`, `shutdown`, `lilo`, `init`.

***/home** : berisi data dari user yang terdaftar dalam komputer / mesin yang bersangkutan.
***/usr** : berisi paket program, dokumentasi, konfigurasi, aplikasi, library dan source aplikasi linux.

***/opt** : berisi aplikasi yang dapat diakses oleh semua user (hampir sama dengan /usr/sbin/.
***/root** : merupakan “home” nya superuser / root / administrator.

***/tmp** : singkatan dari temporer adalah direktori yang disediakan ketika dibutuhkan ruang sementara dalam melakukan pekerjaan, contoh ketika melakukan proses burn cd maka image (file iso) secara default dimasukkan ke direktori ini sebelum di burn ke cd

***/etc** : secara umum merupakan direktori tempat file konfigurasi berbagai macam service dan program yang terinstall di dalam sistem. Direktori /etc berisi file yang berhubungan dengan administrasi system, maintenance script, konfigurasi, security dl. Hanya superuser yang boleh memodifikasi file yang berada di direktori ini. Subdirektori yang sering diakses pada direktori /etc antara lain :

- /etc/profile.d** : berisi skrip yang dijalankan oleh /etc/profile setelah login
- /etc/init.d** : berisi sebagian besar service yang mendukung layanan sistem, misalkan /etc/init.d/networking restart untuk mengaktifkan service jaringan.
- /etc/rc*.d** : dimana tanda “*” berupa angka sesuai dengan default run level. Berisi file untuk service yang dijalankan dan dihentikan pada run level tersebut. Pada sistem berbasis rpm, file ini ter-*symbolic link* ke script inisialisasi sendiri, yang berada pada /etc/rc.d/init.d.
- /etc/skel** : direktori yang berisi beberapa contoh atau kerangka shell inisialisai. Seringkali berisi subdirektori dan file yang digunakan untuk mengisi home directory pengguna baru.
- /etc/X11** :berisi file konfigurasi untuk sistem X Window

Dan beberapa layanan lain diantara nya passwd shadow fstab hosts motd shells services lilo.conf

***/mnt** : berisi informasi device yang terpasang (mount) di dalam komputer.

***/dev** : Berisi file system khusus yang merupakan refleksi device hardware yang dikenali dan digunakan system. Konsep Unix dan Linux adalah memperlakukan peralatan hardware sama seperti penanganan file. Setiap alat mempunyai nama file yang disimpan pada direktori /dev.

<u>Peralatan</u>	<u>Direktori</u>
Floppy	/dev/fd0
Harddisk	IDE : /dev/had, /dev/hdb, /dev/hdc, /dev/hdd SCSI : /dev/sda, /dev/sdb, /dev/sdc
CDROM	SCSI : /dev/scd0, /dev/scd1 IDE : /dev/gscd, /dev/sonycd Universal : /dev/cdrom (link dari actual cdrom ide atau scsi)
Mouse	PS2 : /dev/lp0 Universal : /dev/mouse
Paralel Port	LPT1 : /dev/lp0 LPT2 : /dev/lp1
Serial Port	COM1 : /dev/ttyS0 COM2 : /dev/ttyS1 Universal : /dev/modem (link dari S0 atau S1)

***/var** : Direktori ini berisi data yang bermacam-macam (vary). Perubahan data dalam sistem yang aktif sangatlah cepat. Data-data seperti ini ada dalam waktu yang singkat. Karena sifatnya yang selalu berubah tidak memungkinkan disimpan dalam direktori seperti “/etc”. Oleh karena itu, data-data seperti ini disimpan di direktori var

***/boot** : berisi informasi yang berkaitan dengan device dan service yang dijalankan ketika komputer melakukan booting (proses komputer dari keadaan mati/off menjadi hidup/on)

TIPE FILE

Pada Linux terdapat 6 buah tipe file yaitu

- Ordinary file

- Direktori
- Block Device (Peralatan I/O)

Merupakan representasi dari peralatan hardware yang menggunakan transmisi data per block (misalnya 1 KB block), seperti disk, floppy, tape.

- Character Device (Peralatan I/O)

Merupakan representasi dari peralatan hardware yang menggunakan transmisi data karakter per karakter, seperti terminal, modem, ploter dl

- Named Pipe (FIFO)

File yang digunakan secara intern oleh system operasi untuk komunikasi antar proses

- Link File

PROPERTI FILE

File mempunyai beberapa atribut, antara lain :

- Tipe file : menentukan tipe dari file, yaitu :

<u>Karakter</u>	<u>Arti</u>
-	File biasa
d	Direktori
l	Symbolic link
b	Block special file
c	Character special file
s	Socket link
p	FIFO

- Ijin akses : menentukan hak user terhadap file ini.
- Jumlah link : jumlah link untuk file ini.
- Pemilik (Owner) : menentukan siapa pemilik file ini
- Group : menentukan group yang memiliki file ini
- Jumlah karakter : menentukan ukuran file dalam byte
- Waktu pembuatan : menentukan kapan file terakhir dimodifikasi

- Nama file : menentukan nama file yang dimaksud

Contoh :

```
-rw-rw-r-- 1 bin auth 1639 Oct 31 20:19 /etc/passwd
```

► ► ► ► ► ► ►

pemilik

Tipe	jml link	group	waktu	nama file
izin akses		jml karakter		

NAMA FILE

Nama file maksimal terdiri dari 255 karakter berupa alfanumerik dan beberapa karakter spesial yaitu garis bawah, titik, koma dan lainnya kecuali spasi dan karakter "&", ";", "|", "?", "~", "'", "\"", "[", "]", "(", ")", "\$", "<", ">", "{", "}", "^", "#", "\", "/". Linux membedakan huruf kecil dengan huruf besar (case sensitive). Contoh nama file yang benar :

```
Abcde5434
3
prog.txt
PROG.txt
Prog.txt,old
report_101,v2.0.1
5-01.web.html
```

IJIN AKSES

Setiap obyek pada Linux harus mempunyai pemilik, yaitu nama pemakai Linux (account) yang terdaftar pada /etc/passwd. Ijin akses dibagi menjadi 3 peran yaitu :

- Pemilik (Owner)
- Kelompok (Group)
- Lainnya (Others)

Setiap peran dapat melakukan 3 bentuk operasi yaitu :

- Pada File

R (Read) Ijin untuk membaca

W (Write) Ijin untuk mengubah / membuat

X (Execute) Ijin untuk menjalankan program

- Pada Direktori

R (Read) Ijin untuk membaca daftar file dalam direktori

W (Write) Ijin untuk mengubah/membuat file di direktori

X (Execute) Ijin untuk masuk ke direktori (cd)

Pemilik File/Direktori dapat mengubah ijin akses sebagai berikut :

```
-rwxrwxrwx 1 student test 1639 Oct 31 20:19 file
```

other

group

user

Format untuk mengubah ijin akses

chmod [ugoa] [= + -] [rwx] File(s)

chmod [ugoa] [= + -] [rwx] Dir(s)

dimana u = user (pemilik)

 g = group (kelompok)

 o = others (lainnya)

 a = al

Format lain dari chmod adalah menggunakan bilangan octal sebagai berikut

r	w	x		
4	2	1	=	7

USER MASK

Untuk menentukan izin akses awal pada saat file atau direktori dibuat digunakan perintah umask. Untuk menghitung nilai default melalui umask pada file, maka dapat dilakukan kalkulasi sebagai berikut :

Kreasi file (biasa)	6 6 6
Nilai umask	0 2 2
	----- -
	6 4 4
Kreasi direktori	7 7 7
Nilai umask	0 2 2
	----- -
	7 5 5

SYMBOLIC LINK

Link adalah sebuah teknik untuk memberikan lebih dari satu nama file dengan data yang sama. Bila file asli dihapus, maka data yang baru juga terhapus. Format dari Link :

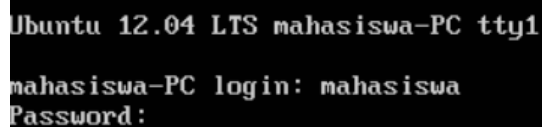
```
ln fileAsli fileDuplikat
```

Symbolic Link diperlukan bila file tersebut di “link” dengan direktori /file yang berada pada partisi yang berbeda. Tipe file menjadi l (link) dan file tersebut menunjuk ke tempat asal. Format :

```
ln -s /FULLPATH/fileAsli /FULLPATH/fileDuplikat
```

L. LANGKAH – LANGKAH

1. Masuk ke sistem operasi Linux. Login : **mahasiswa** password : **mahasiswa**



```
Ubuntu 12.04 LTS mahasiswa-PC tty1
mahasiswa-PC login: mahasiswa
Password: _
```

2. Login ke root

```

Last login: Thu Feb 28 16:13:04 WIT 2013 on tty1
Welcome to Ubuntu 12.04 LTS (GNU/Linux 3.2.0-23-generic-pae i686)

 * Documentation:  https://help.ubuntu.com/

534 packages can be updated.
171 updates are security updates.

mahasiswa@mahasiswa-PC:~$ sudo su
root@mahasiswa-PC:/home/mahasiswa#

```

3. Gunakan perintah **ps** dan **pstree**.
4. Gunakan perintah perintah – perintah berikut :

# cat /proc/cpuinfo	menampilkan informasi CPU
# cat /proc/interrupts	tampilkan penggunaan interupsi
# cat /proc/meminfo	lihat penggunaan memori komputer
# cat /proc/swaps	menampilkan partisi swap yang dipakai linux sebagai memori virtual
# cat /proc/version	tampilkan versi dari kernel.
# cat /proc/net/dev	menampilkan informasi kartu nic dan statistik penggunaan nic

5. Gunakan perintah **ls** untuk melihat tipe file seperti berikut :

# ls	tampilkan berkas-berkas dalam direktori
# ls -F	tampilkan berkas-berkas dalam direktori
# ls -l	menampilkan detil berkas dalam direktori
# ls -a	menampilkan berkas-berkas yang tersembunyi
# ls *[0-9]*	menampilkan berkas-berkas serta direktori yang namanya mengandung angka

6. Gunakan perintah-perintah direktori : **pwd**, **cd**, **mkdir** dan **rmdir** seperti berikut :

# mkdir dir1	membuat sebuah direktori yang diberi nama 'dir1'
# mkdir dir1 dir2	membuat dua direktori dengan satu perintah
# mkdir -p /tmp/dir1/dir2	membuat pohon direktori
# mv dir1 new_dir	merubah nama atau memindahkan direktori dari 'dir1' ke 'new_dir'
# pwd	menunjukkan setapak dari direktori saat ini
# rm -f file1	hapus berkas yang bernama 'file1'.
# rm -rf dir1	menghapus direktori 'dir1' beserta isinya tanpa

	konfirmasi lagi
# rm -rf dir1 dir2	menghapus dua direktori beserta isinya tanpa konfirmasi lagi
# rmdir dir1	menghapus direktori 'dir1'

7. Gunakan perintah-perintah ijin akses : id, grep, chmod, chown, chgrp. Seperti berikut :

# chgrp group1 file1	merubah grup dari data
# chmod ugo+rwX directory1	menetapkan izin membaca (r), menulis (w) dan (x) akses ke pemilik pengguna (u) grup (g) dan lainnya (o)
# chmod go-rwx directory1	memindahkan izin membaca (r), menulis (w) dan (x) akses ke pemilik pengguna (g) dan lainnya (o)
# chown user1 file1	merubah kepemilikan dari data
# chown -R user1 directory1	merubah pengguna pemilik dari direktori dan semua data serta direktori yang ada di dalamnya
# chown user1:group1 file1	Merubah pemilik dan Group dari sebuah File
# grep dhclient /var/log/dmesg	Mencari kata dhclient dalam berkas '/var/log/dmesg'

8. Gunakan perintah user mask : umask.
9. Gunakan perintah link ln seperti berikut

# ln -s file1 lnk1	membuat sebuah tautan simbolis untuk 'file1' ke suatu berkas tautan 'lnk1'
# ln file1 lnk1	membuat tautan fisik antara 'file1' dengan berkas tautan 'lnk1'

M. PERCOBAAN

Percobaan 1 : Melihat ps (process status) dan status direktori /proc

Catatan : Pastikan tidak dalam akses root

1. **ps** menampilkan PID (Process ID) untuk shel dan proses ps itu sendiri

\$ **ps**

\$ ls -l /proc/[Nomor PID]

2. Melihat status proses

\$ cat /proc/[Nomor PID]/status

3. Melihat nilai pada variabel /proc

\$ ls /proc/sys/net/ipv4

4. Melihat isi salah satu variabel

\$ cat /proc/sys/net/ipv4/ip_forward

\$ echo 1 > /proc/sys/net/ipv4/ip_forward (tidak bekerja)

5. Mengubah kernel variable harus dengan ijin akses root. Menjadi root dengan utilitas su (substitute user)

\$ sudo su

[sudo] password for mahasiswa : (masukkan password)

echo 1 > /proc/sys/net/ipv4/ip_forward

exit

6. Kembali ke user semula dan tampilkan variable kernel dengan nilai baru

\$ cat /proc/sys/net/ipv4/ip_forward

Percobaan 2 : Melihat tipe file

1. Melihat block device (peralatan I/O)

\$ ls -l /dev/fd/1

2. Melihat character device (peralatan I/O)

\$ ls -l /dev/tty0

3. Melihat

\$ ls -l /dev/console

4. Melihat direktori

\$ ls -ld /dev

5. Melihat ordinary file

```
$ ls -l /etc/passwd
```

Percobaan 3 : Direktori

1. Melihat direktori HOME

```
$ pwd
```

```
$ echo $HOME
```

2. Melihat direktori aktual dan parent direktori

```
$ pwd
```

```
$ cd ..
```

```
$ pwd
```

```
$ cd ..
```

```
$ pwd
```

3. Membuat satu direktori, lebih dari satu direktori atau sub direktori

```
$ pwd
```

```
$ mkdir A B C A/D A/E B/F A/D/A
```

```
$ ls -l
```

```
$ ls -l A
```

```
$ ls -l A/D
```

4. Menghapus satu atau lebih direktori hanya dapat dilakukan pada direktori kosong dan hanya dapat dihapus oleh pemiliknya kecuali bila diberikan ijin aksesnya

```
$ rmdir B (Terdapat pesan error)
```

```
$ ls -l B
```

```
$ rmdir B/F B
```

```
$ ls -l B
```

5. Navigasi direktori dengan instruksi `cd` untuk pindah dari satu direktori ke direktori lain.

```
$ pwd
$ ls -l
$ cd A
$ pwd
$ cd ..
$ pwd
$ cd /home/mahasiswa/C
$ pwd
$ cd
$ pwd
```

Percobaan 4 : Ijin Akses

1. Melihat identitas diri melalui `etc/passwd` atau `etc/group`

```
$ id
$ grep mahasiswa /etc/passwd
$ grep [Nomor group id] etc/group
```

2. Memeriksa direktori home

```
$ ls -ld /home/mahasiswa
```

3. Mengubah Ijin akses (`chmod`)

```
$ touch file1 file2 file3
$ ls -l
$ chmod u+x file1
$ chmod g=w file1
$ chmod o-r file1
$ ls -l
```

```
$ chmod a=x file2
$ chmod u+x,g-r,o=w file3
$ ls -l
$ chmod 751 file1
$ chmod 624 file2
$ chmod 430 file3
$ ls -l
```

Percobaan 6 : Symbolic Link

1. Link file

```
$ echo "Hallo apa khabar" > halo.txt
$ ls -l
$ ln halo.txt z
$ ls -l
$ cat z
$ mkdir mydir
$ ln z mydir/halo.juga
$ cat mydir/halo.juga
$ ls -l mydir
```

2. Symbolic Link file

```
$ mount
$ ln /home/mahasiswa/z /tmp/halo.txt
$ ln -s /home/mahasiswa/z /tmp/halo.txt
$ ls -l /tmp/halo.txt
$ cat /tmp/halo.txt
```

E. TUGAS

1. Lihat peralatan I/O, character device, yang ada pada system komputer.
2. Buatlah sub direktori **januari**, **februari** dan **maret** sekaligus pada direktori **latihan5**.
3. Buatlah file dataku yang berisi **nama**, **nim** dan **alamat** anda pada sub direktori **januari** dan copy-kan file tersebut ke sub direktori **februari** dan **maret**.
4. Ubahlah ijin akses file dataku pada sub direktori januari sehingga **group** dan others dapat melakukan **write**.
5. Ubahlah ijin akses file dataku pada sub direktori pebruari sehingga user dapat melakukan baik **write**, **read** maupun **execute**, tetapi group dan others hanya bisa read dan execute.
6. Ubahlah ijin akses file dataku pada sub direktori maret sehingga semua dapat melakukan **write**, **read** dan **execute**.
7. Hapuslah direktori **maret**.
8. Ubahkan kepemilikan sub direktori **februari** sehingga **user** dan **group** hanya dapat melakukan **read**, dan cobalah untuk membuat direktori baru **haha** pada sub direktori **februari**.
9. Modifikasi **umask** dari file dataku pada sub direktori **januari** menjadi **027** dan berapakan nilai default-nya ?
10. Buatlah **link** dari file dataku ke file **dataku.ini** dan file **dataku.juga** dan dengan perintah **list** perhatikan berapa **link** yang terjadi ?

LAPORAN HASIL PRAKTIKUM

Nama :

Nim :

Judul Percobaan :

Hasil Percobaan :

Analisis Percobaan :

Kesimpulan Percobaan :

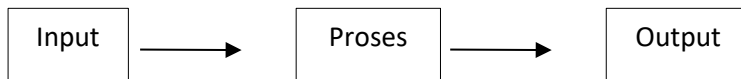
N. TUJUAN

- Mengenal konsep proses I/O dan redirection
- Memahami standar input, output dan error
- Menggunakan notasi output, append dan here document
- Mengenal konsep PIPE dan filter

O.DASAR TEORI

PROSES I/O

Sebuah proses memerlukan Input dan Output.



Instruksi (command) yang diberikan pada Linux melalui Shel disebut sebagai eksekusi program yang selanjutnya disebut proses. Setiap kali instruksi diberikan, maka Linux kernel akan menciptakan sebuah proses dengan memberikan nomor PID (Process Identity).

Proses dalam Linux selalu membutuhkan Input dan menghasilkan suatu Output.

Dalam konteks Linux input/output adalah :

- Keyboard (input)
- Layar (output)
- Files
- Struktur data kernel
- Peralatan I/O lainnya (misalnya Network)

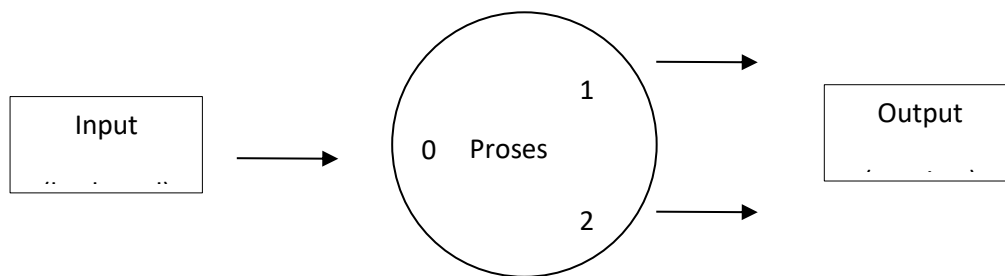
FILE DESCRIPTOR

Linux berkomunikasi dengan file melalui file descriptor yang direpresentasikan melalui angka yang dimulai dari 0, 1, 2 dan seterusnya.

Tiga buah file descriptor standar yang lalu diciptakan oleh proses adalah :

- 0 = keyboard (standar input)
- 1 = layar (standar output)

- 2 = layar (standar error)



Linux tidak membedakan antara peralatan hardware dan file, Linux memanipulasi peralatan hardware sama dengan file.

PEMBELOKAN (REDIRECTION)

Pembelokan dilakukan untuk standard input, output dan error, yaitu untuk mengalihkan file descriptor dari 0, 1 dan 2. Simbol untuk pembelokan adalah :

0< atau < pengganti standard input

1> atau > pengganti standard output

2>

PIPA (PIPELINE)

Mekanisme pipa digunakan sebagai alat komunikasi antar proses.

Input → Proses1 → Output = Input → Proses2 → Output

Proses 1 menghasilkan output yang selanjutnya digunakan sebagai input oleh Proses 2. Hubungan output input ini dinamakan pipa, yang menghubungkan Proses 1 dengan Proses2 dan dinyatakan dengan symbol “|”.

Proses1 | Proses2

FILTER

Filter adalah utilitas Linux yang dapat memproses standard input (dari keyboard) dan menampilkan hasilnya pada standard output (layar). Contoh filter adalah cat, sort, grep, pr, head, tail, paste dan lainnya.

Pada sebuah rangkaian pipa :

P1 | P2 | P3 | Pn-1 | Pn

Maka P2 sampai dengan Pn-1 mutlak harus utilitas Linux yang berfungsi sebagai filter. P1 (awal) dan Pn (terakhir) boleh tidak filter. Utilitas yang bukan filter misalnya who, ls, ps, lp, lpr, mail dan lainnya.

Beberapa perintah Linux yang digunakan untuk proses penyaringan antara lain :

- Perintah grep

Digunakan untuk menyaring masukannya dan menampilkan baris-baris yang hanya mengandung pola yang ditentukan. Pola ini disebut regular expression.

- Perintah wc

Digunakan untuk menghitung jumlah baris, kata dan karakter dari baris-baris masukan yang diberikan kepadanya. Untuk mengetahui berapa baris gunakan option -l, untuk mengetahui berapa kata, gunakan option -w dan untuk mengetahui berapa karakter, gunakan option -c. Jika salah satu option tidak digunakan, maka tampilannya adalah jumlah baris, jumlah kata dan jumlah karakter.

- Perintah sort

Digunakan untuk mengurutkan masukannya berdasarkan urutan nomor ASCII dari karakter.

- Perintah cut

Digunakan untuk mengambil kolom tertentu dari baris-baris masukannya, yang ditentukan pada option -c.

- Perintah uniq

Digunakan untuk menghilangkan baris-baris berurutan yang mengalami duplikasi, biasanya digabungkan dalam pipeline dengan sort.

P. LANGKAH – LANGKAH

10. Masuk ke sistem operasi Linux. Login : **mahasiswa** password : **mahasiswa**

```
Ubuntu 12.04 LTS mahasiswa-PC tty1
mahasiswa-PC login: mahasiswa
Password: _
```

11. Login ke root

```
Last login: Thu Feb 28 16:13:04 WIT 2013 on tty1
Welcome to Ubuntu 12.04 LTS (GNU/Linux 3.2.0-23-generic-pae i686)

 * Documentation:  https://help.ubuntu.com/

534 packages can be updated.
171 updates are security updates.

mahasiswa@mahasiswa-PC:~$ sudo su
root@mahasiswa-PC:/home/mahasiswa#
```

12. Gunakan file descriptor : standard output, standard input dan standard error.
13. Gunakan redirection untuk membelokkan standard output, standard input dan standard error.
14. Gunakan notasi 2>&1, notasi 1>&2 (atau >&2), notasi >>, notasi here document (<<++ ... ++)
dan notasi -.
15. Gunakan pipeline untuk melakukan komunikasi antar proses.

Q. PERCOBAAN

Percobaan 1 : File descriptor

1. Output ke layar (standar output), input dari system (kernel)

```
# ps
```

2. Output ke layar (standar output), input dari keyboard (standard input)

```
# cat
```

```
hallo, apa khabar
```

```
hallo, apa khabar
```

```
(exit dengan ^c[Ctrl-c] )
```

3. Input nama direktori, output tidak ada (membuat direktori baru), bila terjadi error maka tampilan error pada layar (standard error)

```
# mkdir mydir
```

```
# mkdir mydir (Terdapat pesan error)
```

Percobaan 2 : Pembelokan (redirection)

1. Pembelokan standar output

```
# cat 1> myfile.txt
```

Ini adalah teks yang saya simpan

Ke file myfile.txt

2. Pembelokan standar input, yaitu input dibelokkan dari keyboard menjadi dari file

```
# cat 0< myfile.txt
```

```
# cat myfile.txt
```

3. Pembelokan standar error untuk disimpan di file

```
# mkdir mydir (Terdapat pesan error)
```

```
# mkdir mydir 2> myerror.txt
```

```
# cat myerror.txt
```

Logout dari root dan login kembali sebagai mahasiswa

4. Notasi 2>&1 : pembelokan standar error (2>) adalah identik dengan file descriptor 1.

```
$ find /var/spool/cron -print (Terdapat pesan error)
```

```
$ find /var/spool/cron -print 2> out.txt
```

```
$ cat out.txt
```

5. Notasi 1>&2 (atau >&2) : pembelokan standar output adalah sama dengan file descriptor 2 yaitu standar error

```
$ find /var/spool/cron -print 2> outx.txt 1>&2
```

```
$ cat outx.txt
```

6. Notasi >> (append)

```
$ echo "kata pertama" > surat
```

```
$ echo "kata kedua" >> surat
```

```
$ echo "kata ketiga" >> surat
```

```
$ cat surat
```

```
$ echo "kata keempat" > surat
```

```
$ cat surat
```

7. Notasi here document (<<++ ++) digunakan sebagai pembatas input dari keyboard. Perhatikan bahwa tanda pembatas dapat digantikan dengan tanda apa

saja, namun harus sama dan tanda penutup harus diberikan pada awal baris

```
$ cat <<++
Hallo, apa kabar ?
Baik-baik saja ?
Ok!
++

$ cat <<%%%
Hallo, apa kabar ?
Baik-baik saja ?
Ok!
%%%
```

Percobaan 3 : Pipa (pipeline)

```
$ who
$ who | sort
$ who | sort -r
$ who > tmp
$ sort tmp
$ rm tmp
$ ls -l /etc | more
$ ls -l /etc | sort | more
```

Percobaan 4 : Filter

```
$ w -h | grep PM
$ grep st /etc/passwd
$ ls /etc | wc -l
```

```
$ cat > kelas1.txt

Badu

Zulkifli

Yulizir

Yudi

Ade

[Ctrl-c]

$ cat > kelas2.txt

Budi

Gama

Asep

Muchlis

[Ctrl-c]

$ cat kelas1.txt kelas2.txt | sort

$ who | cut -c1-8

$ cat kelas1.txt kelas2.txt > kelas.txt

$ cat kelas.txt | sort | uniq
```

E. TUGAS

1. Lihat daftar secara lengkap pada direktori aktif, belokkan tampilan standard output ke file baru.
2. Lihat daftar secara lengkap pada direktori /etc/paswd, belokkan tampilan standard output ke file baru tanpa menghapus file baru sebelumnya.
3. Urutkan file baru dengan cara membelokkan standard input.
4. Urutkan file baru dengan cara membelokkan standard input dan standard output ke file baru.urut.
5. Buatlah direktori latihan6 sebanyak 2 kali dan belokkan standard error ke file rmdirerror.txt.

6. Urutkan kalimat berikut :

Jakarta

Bandung

Surabaya

Padang

Palembang

Lampung

Dengan menggunakan notasi here document (<@@@ ...@@@)

7. Hitung jumlah baris, kata dan karakter dari file *baru.urut* dengan menggunakan filter dan tambahkan data tersebut ke file *baru*.

8. Gunakan perintah di bawah ini dan perhatikan hasilnya.

```
$ cat /etc/passwd | sort | pr -n | grep tty03
```

```
$ find /etc -print | head
```

```
$ head /etc/passwd | tail -5 | sort
```

9. Gunakan perintah `$ who | cat | cat | sort | pr | head | cat | tail` dan perhatikan hasilnya.

FORMAT LAPORAN HASIL PRAKTIKUM

LAPORAN HASIL PRAKTIKUM	
Nama :	
Nim :	xlvi
Judul Percobaan :	

R. TUJUAN

- Menenal Profile
- Mengerti konsep history
- Membuat dan mengeksekusi shell script sederhana
- Mengerti Job control

S. DASAR TEORI

SHELL

Shell adalah **Command executive**, artinya program yang menunggu instruksi dari pemakai, memeriksa sintak dari instruksi yang diberikan, kemudian mengeksekusi perintah tersebut. Shell ditandai dengan prompt. Untuk pemakai menggunakan prompt \$ dan untuk superuser menggunakan prompt #.

Beberapa macam shell :

- /bin/sh

Bourne shell, dirancang oleh Steve Bourne dari AT&T

- /bin/csh

Dikembangkan oleh UNIX Berkeley yang dikenal dengan C-Shell

- /bin/bash

Kompatibel dengan Bourne Shell dan juga mengadaptasi kemampuan Korn-Shell.

Perbedaan mendasar antara Shell diatas hampir tidak ada, kecuali pada fasilitas pemrograman dan editing.

PROFILE

Pada saat login, program akan menjalankan beberapa program yaitu :

1. /etc/profile

Berisi shell script yang berlaku untuk seluruh pengguna Linux.

2. Profil untuk setiap pemakai

Pada home directory, login pertama kali akan memeriksa file .bash_profile. Bila tidak ada, maka file .bash_login akan dicari. Bila .bash_login tidak ada, maka dicari file bernama .profile.

3. .bashrc

File ini akan dieksekusi untuk perpindahan dari satu shell ke shell yang lain melalui instruksi su.

4. .bash_logout

Pada saat logout, maka bash akan mencari file .bash_logout. Bila ada, file tersebut akan dieksekusi sebelum logout

PATH merupakan daftar nama direktori. Bila sebuah instruksi diberikan dari prompt shell, maka instruksi tersebut akan dicari pada daftar tersebut.

PS1 adalah prompt dimana

\u = Nama User

\h = Nama Host

\W = Nama working directory

HISTORY

History diadaptasi dari C-Shell, yaitu catatan dari semua instruksi yang sejauh ini telah dilakukan. Catatan ini dapat dilihat sebagai history, kemudian dapat dipilih kembali, diedit dan dieksekusi. History memudahkan pemakai untuk mengedit kembali instruksi kompleks dan panjang, terutama bila terjadi kesalahan pada penulisan instruksi maupun parameter.

Navigasi pada daftar history menggunakan karakter kontrol sebagai berikut :

^P (Ctrl-P) melihat instruksi sebelumnya

^N (Ctrl-N) melihat instruksi berikutnya

! eksekusi kembali instruksi sebelumnya

!-3 3 instruksi sebelumnya akan diulang

!88 ulangi instruksi no 88

BASH-SCRIPT

Bash-script adalah file yang berisi koleksi program yang dapat dieksekusi. Untuk eksekusi bash script gunakan . sebelum file bash-script yang berarti eksekusi shell dan tanda ./ berarti file bash-script berada pada direktori actual.

JOB CONTROL

Job adalah sebuah eksekusi program yang diberikan kepada kernel. Sebuah Job dianggap selesai, bila eksekusi program tersebut berakhir. Eksekusi Job adalah sama dengan eksekusi program, baik proses Background maupun proses Foreground

T. LANGKAH – LANGKAH

16. Masuk ke sistem operasi Linux. Login : **mahasiswa** password : **mahasiswa**
17. Melihat profile pada file .bashrc, .bash_history, .bash_logout.
18. Gunakan instruksi history : history dan fc.
19. Buatlah file bash script dengan editor vi; instruksi chmod, &, fg, bg
20. Gunakan perintah job control : jobs
21. Gunakan perintah manipulasi stack untuk direktori : dirs, pushd, popd
22. Gunakan perintah alias dan unalias

U. PERCOBAAN

Percobaan 1 : Profile

1. File .bash_profile dijalankan pada home direktori pemakai yang login. File .bash_profile adalah hidden file, sehingga untuk melihatnya gunakan opsi a pada instruksi ls.

```
$ ls -a
```

```
$ more .bash_profile
```

2. File .bash_logout akan dieksekusi sesaat sebelum logout, berfungsi sebagai house clearing jobs, artinya membersihkan semuanya, misalnya menghapus temporary file atau job lainnya. Melihat file .bash_logout dengan instruksi

```
$ cat .bash_logout
```

Percobaan 2 : History

1. Melihat batasan maksimum instruksi yang dapat disimpan

```
$ ls -l $echo $histsize
```

2. Melihat daftar instruksi yang telah dilakukan dengan perintah history. Daftar instruksi dilengkapi dengan nomor urut. Nomor ini dapat dijadikan parameter dalam mengedit atau mengulang instruksi

```
$ history
```

3. Selain history, instruksi fc (fix command) juga dapat digunakan

```
$ fc -l
```

4. Instruksi `fc` dapat menampilkan instruksi antara 2 nomor atau dengan menggunakan kata depan instruksi yang diberikan

```
$ fc -l [Nomor urut1] [Nomor urut2] (Misalnya fc -l 85 88)
```

Percobaan 3 : Membuat Bash-script dan menjalankannya

1. Membuat file p1.sh

```
$ nano p1.sh  
  
echo "Nama Mahasiswa: Tri Wanda Septian"  
  
echo "Sistem Komputer"
```

2. Mengubah program menjadi executable

```
$ ls -l p1.sh  
  
$ chmod +x p1.sh  
  
$ ls -l p1.sh
```

3. Menjalankan script

```
$ bash p1.sh  
  
$ sh p1.sh  
  
$ . p1.sh  
  
$ ./p1.sh
```

4. Konvensi dalam pembuatan script shell dinyatakan sebagai `#!/bin/bash`.

Tambahkan pada file `p1.sh` konvensi tersebut

```
$ nano p1.sh  
  
#!/bin/bash  
  
echo "Program bash script"
```

5. Buatlah file p2.sh

```
$ nano p2.sh  
  
#!/bin/bash  
  
echo "Program 2 bash script"
```

6. Menjalankan beberapa program shell dalam satu baris instruksi yang dipisahkan dengan tanda `;`

```
$ cat p1.sh ; cat p2.sh
```

```
$ ./p1.sh ; ./p2.sh
```

7. Menjalankan script sebagai proses background, sehingga prompt tidak menunggu program tersebut selesai. Untuk menjalankan proses background gunakan tanda & pada akhir instruksi.

```
$ ./p1.sh &
```

```
$ ./p2.sh &
```

8. Untuk menjalankan 2 program atau lebih dalam satu block, kemudian dieksekusi sebagai proses background, gunakan tanda (...)

```
$ (ls ; who) > hasil &
```

```
$ cat hasil
```

Percobaan 4 : Job Control

1. Proses foreground

```
$ ps x
```

2. Proses background

```
$ ps x > hasil &
```

3. Setiap job mempunyai PID yang tunggal (unique). Untuk melihat jobs yang aktif

```
$ jobs
```

4. Buatlah file ploop.sh. File ini tidak akan pernah berhenti kecuali ditekan Ctrl-C

```
$ nano ploop.sh
```

```
#!/bin/bash
```

```
while [ true ]
```

```
do
```

```
sleep 10
```

```
echo "Hallo"
```

```
done
```

5. Buatlah file ploop.sh menjadi executable. Jalankan program, akan ditampilkan kata Halo setiap 10 detik. Untuk keluar program, tekan Ctrl-C (^C)

```
$ chmod +x ploop.sh
```

```
$ ./ploop.sh
```

6. Jalankan program sebagai background

```
$ ./ploop.sh &
```

7. Periksa job yang aktif

```
$ jobs
```

8. Ubah job menjadi foreground, gunakan fg dan nomor job, atau fg dengan argumen berupa nama jobs yang sedang berjalan, atau bila tidak rancu dengan jobs yang lain, gunakan huruf depan nama jobs yang sedang berjalan.

```
$ fg %[Nomor Job] (Contoh : fg %1) atau
```

```
$ fg %ploop.sh
```

9. Untuk mengembalikan jobs tersebut ke background, tekan Ctrl-Z (^Z), kemudian jalankan instruksi bg

```
$ bg
```

```
$ jobs
```

Percobaan 5 : Manipulasi stack untuk Direktori

1. Instruksi dirs digunakan untuk melihat stack direktori, pada output hanya ditampilkan direktori home ~

```
$ dirs
```

2. Membuat 3 buah direktori

```
$ mkdir marketing sales support
```

3. Masukkan direktori sales ke dalam stack dengan instruksi pushd. Maka terdapat 2 direktori dalam stack yaitu \$HOME/sales dan \$HOME. Kemudian lihat direktori aktual

```
$ pushd sales
```

```
$ pwd
```

4. Masuk ke direktori support

```
$ pushd /home/mahasiswa/support
```

```
$ pwd
```

5. Lakukan kembali untuk direktori marketing

```
$ pushd ../marketing
```

```
$ pwd
```

6. Bila pushd dilakukan tanpa argumen, maka stack akan mengambil direktori berikutnya

```
$ pushd
```

```
$ pushd
```

```
$ pushd
```

7. Untuk membuat direktori sales menjadi direktori paling atas (top stack), maka pushd dapat dilakukan dengan argumen +n, dimana n adalah nomor urut direktori tersebut

```
$ pushd +2
```

```
$ pwd
```

8. Untuk menghapus direktori dari stack, gunakan instruksi popd

```
$ popd
```

```
$ popd +2
```

```
$ dirs
```

Percobaan 6 : Alias

1. Alias adalah mekanisme untuk memberi nama alias pada satu atau sekelompok instruksi. Untuk melihat alias yang sudah terdaftar pada system :

```
$ alias
```

2. Membuat beberapa alias

```
$ alias del='rm -i'
```

```
$ alias h='history'
```

3. Gunakan instruksi hasil alias

```
$ ls
```

```
$ del hasil
```

```
$ h | more
```

4. Untuk menghapus alias gunakan instruksi unalias

```
$ unalias del
```

```
$ del files (Terdapat Pesan Kesalahan)
```

E. TUGAS

1. Eksekusi seluruh profile yang ada :

a. Edit file profile `/etc/profile` dan tampilkan pesan sebagai berikut :

```
echo "Profile dari /etc/profile"
```

b. Asumsi nama anda `stD02001`, maka edit semua profile yang ada yaitu :

```
/home/stD02001/.bash_profile
```

```
/home/. stD02001/.bash_login
```

```
/home/mahasiswa/.profile
```

```
/home/mahasiswa/.bashrc
```

Ganti nama `/home/mahasiswa` dengan nama anda sendiri. Pada setiap

file tersebut, cantumkan instruksi `echo`, misalnya pada `/home/ mahasiswa/.bash_profile` :

```
echo "Profile dari .bash_profile"
```

Lakukan hal yang sama untuk file lainnya, sesuaikan tampilan dengan nama file yang bersangkutan.

c. Jalankan instruksi substitute user, kemudian keluar dengan perintah `exit` sebagai berikut:

```
$ su mahasiswa
```

```
$ exit
```

kemudian gunakan opsi `-` sebagai berikut :

```
$ su - mahasiswa
```

```
$ exit
```

Jelaskan perbedaan kedua utilitas tersebut.

2. Prompt String (PS)

a. Edit file `.bash_profile`, ganti prompt `PS1` dengan `'>'`. Instruksi `export` diperlukan dengan parameter nama variable tersebut, agar perubahan variable `PS1` dikenal oleh semua shell


```
PS1='> '
```

```
export PS1
```

b. Eksperimen hasil PS1 :

```
$ PS1="\! > "
```

```
69 > PS1="\d > "
```

```
Mon Sep 23 > PS1="\t > "
```

```
10:10:20 > PS1="Saya=\u > "
```

```
Saya=mahasiswa > PS1="\w > "
```

```
~ > PS1="\h > "
```

3. Logout

Edit file `.bash_logout`, tampilkan pesan dan tahan selama 5 detik, sebelum eksekusi logout

```
Echo "Terima kasih atas sesi yang diberikan"
```

```
Sleep 5
```

```
clear
```

4. Bash script

a. Buat 3 buah script `p1.sh`, `p2.sh`, `p3.sh` dengan isi masing-masing :

p1.sh

```
#!/bin/bash
```

```
echo "Program p1"
```

```
ls -l
```

p2.sh

```
#!/bin/bash
```

```
echo "Program p2"
```

```
who
```

p3.sh

```
#!/bin/bash
```

```
echo "Program p3"
```

```
ps x
```

b. Jalankan script tersebut sebagai berikut :

```
$ ./p1.sh ; ./p3.sh ; ./p2.sh
```

```
$ ./p1.sh &
```

```
$ ./p1.sh $ ./p2.sh & ./p3.sh &
```

```
$ ( ./p1.sh ; ./p3.sh ) &
```

5. Jobs

a. Buat shell-script yang melakukan loop dengan nama pwaktu.sh, setiap 10 detik, kemudian menyimpan tanggal dan jam pada file hasil.

```
#!/bin/bash
while [ true ]
do
    date >> hasil
    sleep 10
done
```

b. Jalankan sebagai background; kemudian jalankan satu program (utilitas find) di background sebagai berikut :

```
$ jobs
$ find / -print > files 2>/dev/null &
$ jobs
```

c. Jadikan program ke 1 sebagai foreground, tekan ^Z dan kembalikan program tersebut ke background

```
$ fg %1
$ bg
```

d. Stop program background dengan utilitas kill

```
$ ps x
$ kill [Nomor PID]
```

6. History

a. Ganti nilai HISTSIZE dari 1000 menjadi 20

```
$ HISTSIZE=20
```

```
$ h
```

b. Gunakan fasilitas history dengan mengedit instruksi baris ke 5 dari instruksi yang terakhir dilakukan

```
$ !-5
```

c. Ulangi instruksi yang terakhir. Gunakan juga ^P dan ^N untuk bernavigasi pada history bufer

```
$ !!
```

d. Ulangi instruksi pada history bufer nomor 150

```
$ !150
```

e. Ulangi instruksi dengan prefix "ls"

```
$ !ls
```

FORMAT LAPORAN HASIL PRAKTIKUM

LAPORAN HASIL PRAKTIKUM

Nama :

Nim :

lix

Judul Percobaan :

V. TUJUAN

- Mengetahui konsep proses di Linux
- Mengetahui konsep sinyal dan bagaimana cara mengelola sinyal tersebut

W. DASAR TEORI

Setiap kali instruksi diberikan pada Linux shell, maka kernel akan menciptakan sebuah proses-id. Proses ini disebut juga dengan terminology Unix sebagai sebuah Job. Proses Id (PID)

dimulai dari 0, yaitu proses INIT, kemudian diikuti oleh proses berikutnya (terdaftar pada /etc/passwd).

Beberapa tipe proses :

- **Foreground**

Proses yang diciptakan oleh pemakai langsung pada terminal (interaktif, dialog)

- **Batch**

Proses yang dikumpulkan dan dijalankan secara sekuensial (satu persatu). Proses Batch tidak diasosiasikan (berinteraksi) dengan terminal.

- **Daemon**

Proses yang menunggu permintaan (request) dari proses lainnya dan menjalankan tugas sesuai dengan permintaan tersebut. Bila tidak ada request, maka program ini akan berada dalam kondisi “idle” dan tidak menggunakan waktu hitung CPU. Umumnya nama proses daemon di UNIX berakhiran d, misalnya inetd, named, popd dll

SINYAL

Proses dapat mengirim dan menerima sinyal dari dan ke proses lainnya. Proses mengirim sinyal melalui instruksi “kill” dengan format

```
kill [-nomor sinyal] PID
```

Nomor sinyal : 1 s/d maksimum nomor sinyal yang didefinisikan system

Standar nomor sinyal yang terpenting adalah :

NoSinyal	Nama	Deskripsi
1	SIGHUP	Hangup, sinyal dikirim bila proses terputus, misalnya meluiputusnya hubungan modem
2	SIGINT	Sinyal interrupt, melalui ^C
3	SIGQUIT	Sinyal Quit, melalui ^\
9	SIGKILL	Sinyal Kill, menghentikan proses
15	SIGTERM	Sinyal terminasi software

MENGIRIM SINYAL

Mengirim sinyal adalah satu alat komunikasi antar proses, yaitu memberitahukan proses yang sedang berjalan bahwa ada sesuatu yang harus dikendalikan. Berdasarkan sinyal yang

dikirim ini maka proses dapat bereaksi dan administrator/programmer dapat menentukan reaksi tersebut. Mengirim sinyal menggunakan instruksi

```
kill [-nomor sinyal] PID
```

Sebelum mengirim sinyal PID proses yang akan dikirim harus diketahui terlebih dahulu.

X. LANGKAH – LANGKAH

23. Masuk ke sistem operasi Linux. Login : **mahasiswa** password : **mahasiswa**

24. Gunakan instruksi status proses : **ps**.

25. Gunakan instruksi untuk mengelola sinyal : **kill, trap, nohup**

Y. PERCOBAAN

Percobaan 1 : Status Proses

1. Instruksi **ps** (process status) digunakan untuk melihat kondisi proses yang ada. PID adalah Nomor Identitas Proses, TTY adalah nama terminal dimana proses tersebut aktif, STAT berisi S (Sleeping) dan R (Running), COMMAND merupakan instruksi yang digunakan.

```
$ ps
```

2. Untuk melihat factor/elemen lainnya, gunakan option **-u** (user). %CPU adalah presentasi CPU time yang digunakan oleh proses tersebut, %MEM adalah presentasi system memori yang digunakan proses, SIZE adalah jumlah memori yang digunakan, RSS (Real System Storage) adalah jumlah memori yang digunakan, START adalah kapan proses tersebut diaktifkan

```
$ ps -u
```

3. Mencari proses yang spesifik pemakai. Proses diatas hanya terbatas pada proses milik pemakai, dimana pemakai tersebut melakukan login

```
$ ps -u mahasiswa
```

4. Mencari proses lainnya gunakan option **a** (all) dan **au** (all user)

```
$ ps -a
```

```
$ ps -au
```

Percobaan 2 : Sinyal

1. Membuat shell script dengan nama **loop.sh**

```
$ vi loop.sh
```

```
## Sebuah shell script : loop.sh

while [ 1 ]

do

echo ".\c"

sleep 10

done
```

2. Eksekusi file loop.sh sebagai background

```
$ chmod +x loop.sh

$ ./loop.sh &

( tunggu beberapa saat...)
```

3. Melihat proses id

```
$ ps
```

4. Menghentikan proses. Nomor 15 (SIGTERM) merupakan default

```
$ kill -15 [nomor PID] atau

$ kill [nomor PID]
```

5. Menghentikan proses secara mutlak

```
$ kill -9 [nomor PID]
```

Percobaan 3 : Mengelola sinyal

1. Membuat file prog.sh

```
$ vi prog.sh

#!/bin/sh

echo "Program berjalan ..."

while :

do

echo "X"
```

```
sleep 20
```

```
done
```

2. Jalankan program tersebut. Karena program melakukan looping, maka stop dengan mengirim sinyal interrupt (^C)

```
$ chmod +x prog.sh
```

```
$ ./prog.sh
```

3. Jalankan program tersebut sebagai background. Catat nomor PID proses, tekan Enter untuk ke foreground dan periksa melalui instruksi ps

```
$ ./prog.sh &
```

```
$ ps
```

4. Kirimkan sinyal terminasi sebagai berikut

```
$ kill [Nomor PID]
```

5. Ubahlah program prog.sh dengan instruksi trap untuk menangkap sinyal yang dikirim

```
$ vi prog.sh
```

```
#!/bin/sh
```

```
trap " " 1 2 3 15
```

```
echo "Program berjalan ..."
```

```
while :
```

```
do
```

```
echo "X"
```

```
sleep 20
```

```
done
```

6. Jalankan program tersebut sebagai background. Coba lakukan kill dengan nomor PID proses tersebut.

```
$ ./prog.sh &
```

```
$ kill [Nomor PID] atau
```

```
$ kill -1 [Nomor PID] atau
```

```
$ kill -2 [Nomor PID] atau
```

```
$ kill -15 [Nomor PID]
```


7. Perintah `kill` diatas tidak akan menghentikan proses karena dihalangi dengan perintah `trap`. Cobalah menggunakan Nomor sinyal 9

```
$ kill -9 [Nomor PID]
```

E. TUGAS

1. Login sebagai `studentOS` dan lihat status proses, perhatikan kolom keluaran `ps -au` sebagai berikut :

- Sebutkan nama-nama proses yang bukan `root`
- Tulis PID dan `COMMAND` dari proses yang paling banyak menggunakan CPU time
- Sebutkan buyut proses dan PID dari proses tersebut
- Sebutkan beberapa proses `daemon`
- Pada prompt login lakukan hal-hal sebagai berikut :

```
$ csh
```

```
$ who
```

```
$ bash
```

```
$ ls
```

```
$ sh
```

```
$ ps
```

Sebutkan PID yang paling besar dan kemudian buat urutan proses sampai ke `PPID =1`. Lakukan `^d` atau `exit` atau `logout` sampai kembali muncul login: prompt

2. Modifikasi program `prog.sh` sebagai berikut :

```
$ vi prog.sh
```

```
#!/bin/sh
```

```
trap "echo Hello Goodbye ; exit 0" 1 2 3 15
```

```
echo "Program berjalan ..."
```

```
while :
```

```
do
```

```
echo "X"

sleep 20

done
```

Jalankan program tersebut sebagai background. Coba lakukan kil dengan nomor sinyal 1, 2, 3 dan 15 pada nomor PID proses tersebut. Apakah proses berhenti atau tetap berjalan? Nomor sinyal berapa yang digunakan untuk menghentikan proses diatas?

3. Modifikasi program

myjob.sh. Buatlah trap sedemikian rupa, sehingga bila proses tersebut dihentikan (kil), otomatis file berkas akan terhapus.

```
$ vi myjob.sh

#!/bin/sh

trap _____

i=1

while :

do

    find / -print > berkas

    sort berkas -o hasil

    echo "Proses selesai pada `date`" >> proses.log

    sleep 60

done

$ kill -15 [Nomor PID]

$ ls -l
```

FORMAT LAPORAN HASIL PRAKTIKUM

LAPORAN HASIL PRAKTIKUM

Nama :

Nim :

Judul Percobaan :

Hasil Percobaan :

Z. TUJUAN

- Mengenal utilitas dasar Linux dan Unix
- Merangkaikan utilitas dengan pipe
- Mempelajari konsep delimiter

AA. DASAR TEORI

SORTIR

Sortir dengan utilitas sort akan menyusun data berdasarkan criteria.

Utilitas sort dilakukan dengan format

sort option file(s)

Adapun option yang dapat diberikan :

- -r (reverse) yaitu menyusun terbalik dari Z ke A
- $\pm f.c$ yaitu penyusunan berdasarkan pointer yang diperintahkan. f adalah

nomor field, c adalah nomor karakter

Contoh : +2 berarti pointer berada setelah 2 field

+2.3 berarti pointer berada setelah 2 field + 3 karakter

-2 berarti pointer berada sebelum 2 field

-2.5 berarti pointer berada sebelum 2 field + 5 karakter

- -n yaitu komparasi untuk menyusun berdasarkan nilai numeric.
- -t yaitu sebagai pemberitahuan suatu pemisah (delimiter) dari suatu table.

PREFORMAT

Instruksi pr (preformat) digunakan untuk menyusun data sebelum dicetak ke printer. Instruksi pr akan menyiapkan header, nomor halaman dan lainnya. Opsi untuk instruksi pr antara lain :

- -n : menampilkan nomor baris
- -Nr : menampilkan nomor halaman dimulai dengan Nr
- -t : tidak menggunakan header atau trailer
- -h teks : menampilkan teks sebagai judul
- -l jml : jumlah baris dalam jml, default adalah 66 baris

WORD COUNT

Utilitas wc (word count) digunakan untuk

- -c : menghitung jumlah karakter (chars)
- -w : menghitung jumlah kata (words)
- -l : menghitung jumlah baris (lines)

Utilitas `wc` banyak digunakan untuk pemrograman shel, terutama untuk `sysadmin`, misalnya melihat jumlah pemakai yang terdaftar di `etc/passwd` atau melihat jumlah group dalam system.

TRANSLATE

Utilitas `tr` (translate) mengubah pengulangan karakter menjadi karakter yang lain. Utilitas `tr` akan membaca input dari standard input, kemudian mengubahnya dari satu pola ke karakter yang lain. Opsi untuk instruksi `tr` antara lain :

- `-s` : menghapus pengulangan (squeeze repeats)
- `-d` : menghapus karakter tertentu

MEMOTONG KARAKTER

Utilitas `cut` (memotong karakter) digunakan untuk memotong karakter dari sebuah kata dalam baris. Opsi untuk instruksi `cut` antara lain :

- `-f` : memenggal berdasarkan posisi field. Opsi ini hanya akan berhasil bila antara kata dipisahkan dengan delimiter (separator) yaitu tabulator. Bila delimiter bukan tab, maka opsi `-d` harus ditambahkan sebagai info tentang delimiter yang dimaksud.
- `-c` : memenggal berdasarkan posisi karakter.

HEAD DAN TAIL

Utilitas `head` akan menampilkan isi file dari awal hingga akhir, sesuai dengan opsi yang diberikan. Utilitas `tail` akan menampilkan isi file dari akhir hingga awal, sesuai dengan opsi yang diberikan. Utilitas ini sangat bermanfaat terutama untuk menampilkan awal atau akhir dari satu bagian file yang besar.

FIND

Utilitas `find` digunakan untuk menemukan file dengan criteria tertentu. Utilitas ini banyak digunakan untuk melokalisasi file dan kepentingan backup.

Sintak dari utilitas `find` :

```
find [daftar direktori] [ekspresi]
```

Opsi dari utilitas `find` :

- `-print` : menampilkan hasil pencarian ke standard output
- `-name [Namafile]` : mencari file dengan nama yang sama dengan [Namafile]
- `-type [Tipefile]` : mencari file dengan tipe file seperti ordinary (-), directory (d), pipe, socket (s), block device (b), character device (c) dll

- `-links ±n` : mencari file dengan jumlah link sama dengan n, lebih besar atau lebih kecil dari n
- `-user [NamaUser]` : mencari file dengan nama user sama dengan nama yang terdaftar di `/etc/passwd`
- `-group [NamaGroup]` : mencari file dengan nama group yang sama dengan nama yang terdaftar di `/etc/group`
- `-perm [Mode]` : mencari file dengan ijin akses tertentu
- `-inum [Nomor]` : mencari file dengan nomor inode tertentu. Setiap file mempunyai nomor inode yang dapat dilihat dengan opsi `-l` pada instruksi `ls`
- `-size ±n[c|k]` : mencari file dengan jumlah karakter (c) tertentu atau dalam kilobyte (b)
- `-atime ±n` : mencari file yang terakhir diakses pada jumlah hari tertentu
- `-mtime ±n` : mencari file yang terakhir dimodifikasi pada jumlah hari tertentu
- `-newer [NamaFile]` : mencari file yang berusia lebih baru dari [NamaFile]
- `-exec [command]\` : bila criteria terpenuhi, maka command akan dieksekusi
- `-ok [command]\` : sama dengan `exec`, kecuali `find` akan memberikan pertanyaan yes atau no untuk mengeksekusi command tersebut.
- `-depth` : mengolah subdirektori terlebih dahulu
- `-nouser` : mencari file yang pemiliknya tidak lagi terdaftar pada `/etc/passwd`
- `-nogroup` : mencari file yang groupnya tidak lagi terdaftar pada `/etc/group`

BB. LANGKAH – LANGKAH

26. Masuk ke sistem operasi Linux. Login : **mahasiswa** password : **mahasiswa**

27. Gunakan utilitas linux : `sort`, `pr`, `wc`, `tr`, `cut`, `head`, `tail`, `find`.

28. Gunakan pipe dan delimiter pada utilitas tersebut.

CC. PERCOBAAN

Percobaan 1 : Sortir

1. Buatlah file `mobil.db` sebagai berikut

(note: jangan menggunakan tombol tab, tapi spasi, untuk memisahkan kolom)

```
$ cat > mobil.db
```

Badu	Honda	Prelude	150.000.000
Hasan	Toyota	Kijang	125.450.000
Adam	BMW	320i	191.000.000
Zoros	Toyota	Kijang	116.000.000
Stefan	Peugeot	405	288.654.000
Andriane	Opel	Blazer	186.500.000

Ctrl-D (^D)

2. Lakukan proses sorting pada file mobil.db

```
$ sort mobil.db
```

3. Lakukan proses sorting dengan susunan terbalik

```
$ sort -r mobil.db
```

4. Lakukan proses sorting berdasarkan karakter 2 pada field pertama

```
$ sort +0.1 mobil.db
```

5. Lakukan proses sorting berdasarkan manufaktur mobil

```
$ sort +1 mobil.db
```

6. Lakukan proses sorting berdasarkan manufaktur dan nama pemilik mobil

```
$ sort +1 -2 mobil.db
```

7. Lakukan proses sorting berdasarkan nilai numeric (yaitu field ke 4)

```
$ sort -n +3 mobil.db
```

8. Lakukan proses sorting berdasarkan harga termahal

```
$ sort -nr +3 mobil.db
```

Percobaan 2 : Separator

1. Sebuah table umumnya dipisahkan dengan spasi atau tabulator, tetapi bisa juga dengan menggunakan tanda koma untuk memisahkan kolom

```
$ cat > peserta
```

```
Anjas Asmara,NT Full Package,Jakarta
```

```
Shamir Gwindani,Unix Advanced,Bandung
```

```
Shakila,ASP.NET,Yogya
```

Agustin Rosa,VB.NET,Bali

Imelda Pora,Cisco Routing,Jakarta

Sabar Sobar,Linux Network,Bandung

^D

2. Lakukan proses sorting

```
$ sort peserta
```

3. Menyusun berdasarkan kota, hal ini tidak dapat dilakukan. Agar sort mengerti pemisah (delimiter) adalah koma, maka harus diberikan opsi -t

```
$ sort +2 peserta
```

```
$ sort -t, +2 peserta
```

Percobaan 3 : Preformat

```
$ pr mobil.db|more
```

```
$ pr -h "Daftar Pemilik Mobil" -n mobil.db|more
```

Percobaan 4 : Word Count

```
$ wc mobil.db
```

```
$ wc -l mobil.db
```

```
$ wc -w mobil.db
```

```
$ wc -c mobil.db
```

Percobaan 5 : Paste untuk menggabungkan 2 atau lebih file secara vertikal

```
$ cat > fileA
```

```
aaaaa
```

```
bbbbb
```

```
cccc
```

```
$ cat > file1
```

```
11111
```

```
22222
```

```
33333
```

```
44444
```



```
55555
```

```
$ paste fileA file1
```

```
$ paste fileA file1 > fileX
```

Percobaan 6 : Translate

1. Mengubah huruf a yang diinputkan dari keyboard menjadi X

```
$ tr 'a' 'X'
```

```
apa
```

```
XpX
```

```
Khabar
```

```
khXbXr
```

```
^D
```

2. Mengubah semua huruf kecil menjadi huruf besar dan sebaliknya

```
$ cat mobil.db|tr '[a-z]' '[A-Z]'
```

```
$ cat mobil.db|tr '[A-Z]' '[a-z]'
```

3. Menghapus huruf a yang berulang dan diganti dengan 1 huruf a saja

```
$ tr -s 'a' 'a'
```

```
apaa khaaaaaabaaar
```

```
apa khabar
```

4. Menghapus spasi pada file mobil.db

```
$ cat mobil.db | tr -s ' ' '\n'
```

5. Enkripsi file sederhana dengan mengacak huruf

```
$ cat > to-pacar
```

```
Kepada pacar saya,
```

```
Jangan lupa nonton di plasa tunjungan
```

```
Jam 5 ketemu di rumah saya ya
```

```
Pacar kamu
```

`^D`

```
$ cat to-pacar | tr '[A-M] [N-Z] [a-m] [n-z]' '[N-Z] [A-M] [n-z] [a-m]' > secret-mail  
$ cat secret-mail
```

6. Mengembalikan enkripsi file dengan cara yang sama secara terbalik

```
$ cat secret-mail | tr '[N-Z] [A-M] [n-z] [a-m]' '[A-M] [N-Z] [a-m] [n-z]'
```

7. Menghilangkan karakter tertentu

```
$ cat to-pacar | tr -d 'a'  
$ cat to-pacar | tr -d '\n'
```

Percobaan 7 : Memotong karakter

1. Mengambil field ke 2 dari file mobil.db dengan terlebih dahulu menghilangkan pengulangan spasi terlebih dahulu

```
$ cat mobil.db|tr -s ' ' '  
$ cat mobil.db|tr -s ' ' '\|cut -d' ' -f2
```

2. Mengurut output

```
$ cat mobil.db|tr -s ' ' '\|cut -d' ' -f2|sort
```

3. Menghilangkan baris duplikasi

```
$ cat mobil.db|tr -s ' ' '\|cut -d' ' -f2|sort|uniq
```

4. Memotong karakter posisi ke 3 sampai dengan posisi ke 5 dari nama pemilik mobil

```
$ cut -c3-5 mobil.db
```

5. Kombinasi memotong field dengan koma

```
$ cat mobil.db|tr -s ' ' '\|cut -d' ' -f2,4
```

6. Menggunakan Tab yang diselipkan sebagai delimiter untuk tampilan lebih baik

```
$ cat mobil.db|tr -s ' ' '\|cut -d' ' -f2,4|tr ' ' '\t'
```

Percobaan 8 : Head dan Tail

1. Mengambil 3 baris dari awal (head) dan 3 baris terakhir (tail)

```
$ cat mobil.db
```

```
$ head -3 mobil.db
```

```
$ tail -3 mobil.db
```

Percobaan 9 : Find

1. Menampilkan semua file yang ada di curen direktori

```
$ find . -print
```

2. Mencari file paswd di direktori /etc, /lib dan /usr/bin

```
$ find /etc /lib /usr/bin -name passwd
```

3. Mencari file group pada root direktori. Karena bukan superuser, banyak ijin akses ditolak. Untuk membuangnya gunakan descriptor 2>

```
$ find / -name group
```

```
$ find / -name group 2>/dev/null
```

4. Mencari file dengan tipe pipe

```
$ find / -type p 2>/dev/null
```

5. Mencari socket di system file

```
$ find / -type s 2>/dev/null
```

6. Mencari jumlah link 6

```
$ find /lib -links 6
```

7. Mencari nama user mahasiswa

```
$ find / -user mahasiswa 2>/dev/null
```

8. Mencari nama group mahasiswa

```
$ find /tmp -group mahasiswa 2>/dev/null
```

9. Mencari ijin akses 777 pada root direktori

```
$ find / -perm 777 2>/dev/null
```

10. Mencari file yang berukuran 4K, lebih besar dari 4K dan lebih kecil dari 4K

```
$ find . -size 4k
```

```
$ find . -size +4k
```

```
$ find . -size -4k
```

12. Mencari file yang terakhir diakses pada satu hari sebelumnya

```
$ find /home -atime -1
```

13. Mencari file yang terakhir dimodifikasi dalam satu hari

```
$ find /home -mtime +1
```

14. Mencari file yang lebih baru dari file x

```
$ touch x
```

```
$ find . -newer x
```

```
$ touch y
```

```
$ find . -newer c
```

15. Melihat isi direktori bila file x ditemukan. Tanda {} merepresentasikan nama file yang ditemukan

```
$ find -name x
```

```
$ find -name x -exec ls -l {} \;
```

16. Menghapus file x bila ditemukan

```
$ find -name x -ok rm {} \;
```

E. TUGAS

1 Jelaskan tujuan dari perintah berikut :

```
$ wc -l /etc/passwd
```

```
$ wc -l /etc/group
```

2. Buatlah file status dan gabungkan file ini dengan mobil

```
$ cat > status
```

```
-
```

```
-
```

```
dijual
```

```
-
```

dijual

-

3. Gunakan utilitas translate untuk mengganti seluruh huruf hidup dari teks nyanyian berikut dengan huruf o semua

```
$ cat > burung
burung kakak tua
hinggal di jendela
nenek sudah tua
giginya tinggal dua
^D
```

4. Periksa /etc/paswd dan ambil field ke 5 dengan perintah cut. Jangan lupa mencantumkan delimiter yang berupa tanda ':'.
5. Apa maksud dari perintah berikut :

```
who | cut -c 1|sort|uniq|wc
```

FORMAT LAPORAN HASIL PRAKTIKUM

LAPORAN HASIL PRAKTIKUM

Nama :

Nim :

Judul Percobaan :

Hasil Percobaan :

DD. TUJUAN

- Mengetahui variable dan operasi assignment.
- Menggunakan struktur if – fi
- Menggunakan struktur case – esac.
- Loop dengan while, until, for, do while.
- Membuat fungsi dan mengetahui cara memanggil fungsi tersebut.

EE. DASAR TEORI

VARIABLE DAN OPERASI ASSIGNMENT

Variable merupakan tempat menyimpan data yang akan diproses. Operasi pemasukan data ke dalam variable dinamakan dengan assignment. Proses assignment menggunakan tanda “=”. Kita dapat menggunakan perintah **let** sebagai pernyataan perintah untuk melakukan proses assignment. Perintah **let** memungkinkan kita menjadi lebih leluasa untuk menyatakan perintah assignment.

PERINTAH **expr** DAN **let**

Perintah **expr** merupakan perintah yang digunakan untuk melakukan suatu evaluasi suatu ekspresi (pernyataan), umumnya operasi pengolahan data, yang dapat melibatkan operator di dalamnya.

Perintah **let** merupakan perintah yang digunakan untuk melakukan proses assignment hasil suatu ekspresi operasi, aritmatika, string, dan operasi lainnya ke dalam variable.

Berbeda dengan perintah **expr** yang melakukan evaluasi dan menampilkan hasilnya ke standar output, hasilnya tidak disimpan dalam variable, sedangkan perintah **let** akan disimpan ke dalam variable, tetapi hasilnya tidak ditampilkan ke standar output

KONSTRUKSI IF

Konstruksi pemilihan merupakan suatu blok urutan perintah yang dibentuk dengan menggunakan perintah yang diawali dengan **if** dan diakhiri dengan **fi**. Pemeriksaan suatu kondisi akan menghasilkan benar atau salah, bila kondisi terpenuhi (bernilai benar) maka seurutan perintah yang mengikuti **if** ini akan dikerjakan.

```
if kondisi
then
    perintah
fi
```

Nilai suatu kondisi bisa jadi mengakibatkan ada dua alternatif urutan perintah yang harus dikerjakan, yaitu perintah jika kondisi benar dan perintah jika kondisi salah.

```
if kondisi
then
    perintah jika kondisi benar
else
    perintah jika kondisi salah
```

fi

Jika hasil pemeriksaan memiliki lebih dari dua kondisi maka kita harus membuat pemeriksaan yang kedua di dalam blok else, demikian juga apabila ada kondisi lain yang mungkin ada, pemeriksaan kondisi yang baru tersebut dilakukan di dalam else dari blok if yang ada dalam else yang pertama.

if kondisi

then

if kondisi

then

perintah

else

perintah

fi

else

if kondisi

then

perintah

else

perintah

fi

fi

Dalam pemrograman shell disediakan bentuk yang menyederhanakan proses ini, dengan menggunakan **elif**.

if kondisi

then

perintah

elif kondisi

then

perintah


```

else
    perintah
fi

```

KONSTRUKSI CASE

Case digunakan untuk menyederhanakan pemakaian if yang berantai, sehingga dengan case, kondisi dapat dikelompokkan secara logis dengan lebih jelas dan mudah untuk ditulis.

```

case variable in
    match1)
        instruksi1.1
        instruksi1.2
        .....
;;
    match2)
        instruksi2.1
        instruksi2.2
        .....
;;
    *)
        instruksi3.1
        instruksi3.2
        .....
;;
esac

```

Case diakhiri dengan esac dan pada setiap kelompok instruksi diakhiri dengan ;. Pada akhir pilihan yaitu *) yang berarti adalah “default”, bila kondisi tidak memenuhi pola sebelumnya

KONSTRUKSI FOR

For digunakan untuk pengulangan dengan menggunakan var yang pada setiap pengulangan akan diganti dengan nilai yang berada pada daftar (list).

```

for var in str1 str2 .....strn

```

```
do
    instruksi1
    instruksi2
    .....
done
```

KONSTRUKSI WHILE

While digunakan untuk pengulangan instruksi, yang umumnya dibatasi dengan suatu kondisi. Selama kondisi tersebut TRUE, maka pengulangan terus dilakukan. Loop akan berhenti, bila kondisi FALSE, atau program keluar dari blok while melalui exit atau break.

```
while kondisi
do
    instruksi1
    instruksi2
    .....
done
```

INSTRUKSI DUMMY

Instruksi dummy adalah instruksi yang tidak melakukan apa-apa, namun instruksi ini memberikan status exit 0 (TRUE). Oleh karena itu, instruksi dummy dapat digunakan sebagai kondisi forever pada loop (misalnya while).

Simbol instruksi dummy adalah → :

FUNGSI

Fungsi adalah program yang dapat dipanggil oleh program lainnya dengan menggunakan notasi NamaFungsi(). Fungsi memberikan exit status (\$?) yang dinyatakan dengan **return nr**, atau nilai 0 sebagai default. Membuat fungsi diawali dengan nama fungsi, parameter, kemudian blok program yang dinyatakan dalam { ... }.

Contoh :

```
F1( ) {
    .....
    .....
    return 1
}
```

Variabel dapat didefinisikan dalam fungsi sebagai variable local atau global. Hal yang perlu diperhatikan, nama variable yang digunakan dalam sebuah fungsi, jangan sampai bentrok dengan nama variable yang sam ada luar fungsi, sehingga tidak terjadi isi variable berubah.

FF. LANGKAH – LANGKAH

29. Masuk ke sistem operasi Linux. Login : **mahasiswa** password : **mahasiswa**

30. Lakukan operasi assignment.

31. Gunakan instruksi if - fi

32. Gunakan instruksi case-esac.

33. Gunakan instruksi looping for-do-done, while-do-done, until, dummy.

34. Buatlah fungsi dengan variable local dan global.

GG. PERCOBAAN

Percobaan 1 : Variable dan operasi assignment

1. Ketik variable-variable ini setelah prompt

```
$NAMA="Ferry"
```

```
$ANGKA=100
```

```
$NAMALENGKAP='Tri Wanda Septian'
```

```
$SIAPASAJA=`who`
```

2. Assignment dengan Let. Ketik assignment di bawah ini setelah prompt

```
$let a=10
```

```
$let "a=10"
```

```
$let a=5+6
```

```
$let a= 5 + 6 (salah)
```

```
$let "a=5+6"
```

3. Variable read only. Ketik setelah prompt

```
$NAMA="Faiz"
```

```
$readonly NAMA  
$echo $NAMA  
$NAMA="Muhammad"
```

Percobaan 2 : Menampilkan isi variable

1. Buatlah file prog01.sh dengan editor vi

```
$vi prog01.sh  
NAMA="FASILKOM"  
KODEPOS=30256  
SIAPA_SAJA=`who`  
echo "isi variable NAMA" $NAMA  
echo "isi variable KODEPOS" $KODEPOS  
echo "isi variable SIAPA_SAJA" $SIAPA_SAJA
```

2. Jalankan program prog01.sh

```
$ . prog01.sh
```

3. Buatlah file prog02.sh dengan editor vi

```
$vi prog02.sh  
NAMA="Selly marisca"  
echo `isi variabel NAMA $NAMA`  
echo "isi variabel NAMA \"$NAMA"
```

4. Jalankan program prog02.sh

```
$ . prog02.sh
```

5. Buatlah file prog03.sh dengan editor vi

```
$vi prog03.sh  
let a=10  
echo $a  
let "a=10"
```

```
echo $a  
let a=5+6  
echo $a  
let a = 5  
echo $a  
let "a = 5"  
echo $a
```

6. Jalankan program prog03.sh

```
$ . prog03.sh
```

Percobaan 3 : Menerima Masukan

1. Buatlah file prog04.sh dengan editor vi, kemudian jalankan

```
echo "Program untuk menerima masukan"  
echo "=====  
echo "Masukkan nama Anda :"  
read NAMA  
echo "welcome $NAMA dalam shell script programming"
```

Percobaan 4 : expr dan let

1. Ketik setelah prompt sintaks di bawah ini

```
$expr 5  
$expr 5 + 6
```

2. Ketik setelah prompt sintaks di bawah ini

```
$echo "Hallo"  
$expr "Hallo"  
$echo 5 + 6
```

```
$expr 5 + 6
```

```
$expr 5+6
```

3. Ketik setelah prompt sintaks di bawah ini

```
$expr 5 + 6
```

```
$let x=5+6
```

```
$echo $x
```

```
$y=`expr 5 + 6 `
```

```
$echo $y
```

```
$expr $y
```

```
$expr $x
```

Percobaan 5 : Operasi aritmatika

1. Buatlah file prog05.sh dengan editor vi, kemudian jalankan

```
$vi prog05.sh
```

```
#!/bin/sh
```

```
echo "Program penjumlahan dan perkalian dua bilangan"
```

```
echo "Bilangan I ?"
```

```
read bil1
```

```
echo "Bilangan II ?"
```

```
read bil2
```

```
jml=`expr $bil1 + $bil2`
```

```
kali=`expr $bil1 \* $bil2`
```

```
echo "hasil penjumlahan $bil1 + $bil2 = $jml"
```

```
echo "hasil perkalian $bil1 * $bil2=$kali"
```

Percobaan 6 : Konstruksi if – fi

1. Buatlah file prog06.sh dengan editor vi, kemudian jalankan

```
$vi prog06.sh
```

```
echo "Utilitas membuat direktori baru"
```

```

echo "masukkan nama direktori baru yang akan dibuat ?"
read NAMADIR
mkdir $NAMADIR
if test $? -eq 0
then
    echo "Direktori dengan nama $NAMADIR telah berhasil
dibuat"
fi

```

2. Buatlah file prog07.sh dengan editor vi, kemudian jalankan

```

$vi prog07.sh

    echo "Utilitas membuat direktori baru"
echo "masukkan nama direktori baru yang akan dibuat ?"
read NAMADIR
mkdir $NAMADIR 2> /dev/null
if ( $? -eq 0 )
    then
        echo "direktori dengan nama $NAMADIR telah berhasil
dibuat"
    else
        echo "direktori dengan nama $NAMADIR gagal dibuat"
        echo "mgk direktori $NAMADIR sdh ada"
    fi

```

3. Konstruksi pemilihan dengan if dan test tanpa \\$. Buatlah file prog08.sh dengan editor vi. Kemudian jalankan

```

$vi prog08.sh

echo "konstruksi pemilihan dengan if dan test tanpa \$"
if test 5 -lt 6
then

```

```
echo "5 lbh kecil dari 6"
else
    echo "5 tdk lebih kecil dari 6"
fi
```

4. Buatlah file prog09.sh dengan editor vi. Kemudian jalankan

```
$vi prog09.sh
echo "konstruksi pemilihan dengan if dan test tanpa \$"
if test 6 -gt 5
then
echo "5 lbh kecil dari 6"
else
    echo "5 tdk lebih kecil dari 6"
fi
```

5. Alias perintah test adalah tanda "[" dan "]".Buatlah file prog10.sh dengan editor vi. Kemudian jalankan

```
$vi prog10.sh
echo "Menentukan bil terbesar"
echo "Masukkan bil pertama ?"
read bil1
echo "Masukkan bil kedua ?"
read bil2
if [ $bil1 -gt $bil2 ]
then
    echo "$bil1 lebih besar dari $bil2"
else
    echo "$bil2 lebih besar dari $bil"
fi
```


6. Buatlah program untuk membandingkan 2 buah bilangan, apakah lebih besar, lebih kecil atau sama dengan menggunakan if bersarang (nested if). Simpan dengan nama file prog11.sh. Kemudian jalankan

7. Sempurnakan program di bawah ini :

```
        echo "Operasi Aritmatika dengan Menu"
echo "1) Penjumlahan "
echo "2) Selisih "
echo "3) Perkalian"
echo "4) Pembagian "
echo "Nomor Pilihan : "
read pilihan
echo "Masukkan dua bilangan "
echo "Bilangan pertama : "
read bil1
echo "Bilangan kedua : "
read bil2
if [ $pilihan -eq 1 ]
then
        let "hasil=$bil1 + $bil2
elif .....
then

.....
fi

        echo "Hasil = $hasil"
```

Percobaan 7 : Konstruksi case - esac

1. Buatlah file prog14.sh dengan editor vi

```
$ vi prog14.sh
```

```
#!/bin/sh

# Prog: prog14.sh

echo "1. Siapa yang aktif"
echo "2. Tanggal hari ini"
echo "3. Kalender bulan ini"
echo -n "    Pilihan : "
read PILIH

case $PILIH in
1)
    echo "Yang aktif saat ini"
    who
    ;;
2)
    echo "Tanggal hari ini"
    date
    ;;
3)
    echo "Kalender bulan ini"
    cal
    ;;
*)
    echo "Salah pilih !!"
    ;;
esac
```

2. Jalankan program prog14.sh, cobalah beberapa kali dengan inputan yang berbeda

```
$ . prog14.sh
```

3. Buatlah file prog15.sh yang merupakan bentuk lain dari case

```

$ vi prog15.sh

#!/bin/sh

# Prog: prog15.sh

echo -n "Jawab (Y/T) : "

read JWB

case $JWB in
y | Y | ya |Ya |YA ) JWB=y ;;
t | T | tidak | Tidak | TIDAK ) JWB=t ;;
esac

```

4. Jalankan program prog15.sh, cobalah beberapa kali dengan inputan yang berbeda

```
$ . prog15.sh
```

5. Modifikasi file prog15.sh yang merupakan bentuk lain dari case

```

$ vi prog15.sh

#!/bin/sh

# Prog: prog15.sh

echo -n "Jawab (Y/T) : \c"

read JWB

case $JWB in
[yY] | [yY][aA] ) JWB=y ;;
[tT] | [tT]idak ) JWB=t ;;
*) JWB=? ;;
esac

```

6. Jalankan program prog15.sh, cobalah beberapa kali dengan inputan yang berbeda

```
$ . prog15.sh
```

7. Buatlah program Aritmatika dengan menu dengan menggunakan struktur case

Percobaan 8 : Konstruksi for-do-done

1. Buatlah file prog16.sh dengan editor vi, kemudian jalankan

```
$vi prog16.sh

echo "Menampilkan nama dengan for"
a=0

for item in Budi Tuti Dian Rudi
do

    let a+=1

    echo "item no $a adalah $item"

done
```

2. Buatlah file prog17.sh dengan editor vi, kemudian jalankan

```
$vi prog17.sh

echo "file yg pernah dibuat"
for namafile in *.sh
do

echo "nama file script : $namafile "      done
```

3. Buatlah file prog18.sh dan jalankan

```
$vi prog18.sh

#!/bin/sh

# Prog: prog18.sh

for NAMA in bambang harry kadir amir
do

echo "Nama adalah : $NAMA"

done
```

4. Buatlah file prog19.sh yang berisi konstruksi for dan wildcard, program ini akan menampilkan nama file yang berada di curen direktori

```
$vi prog19.sh
```

```
#!/bin/sh

# Prog: prog12.sh
```

```
for F in *
do
echo $F
done
```

5. Modifikasi file prog19.sh, program ini akan menampilkan long list dari file yang mempunyai ekstensi lst

```
$ vi prog19.sh

#!/bin/sh

# Prog: prog19.sh
```

```
for F in *.lst
do
ls -l $F
done
```

Percobaan 9 : Konstruksi while-do-done

1. Buatlah file prog20.sh dengan editor vi, kemudian jalankan

```
$vi prog20.sh

i=0
while [ i$ -lt 10 ]
do
    echo ""
    let i+=1
done
```

2. Buatlah file prog21.sh dengan editor vi, kemudian jalankan

```
$vi prog21.sh

#!/bin/sh

# Prog: prog21.sh

PILIH=1

while [ $PILIH -ne 4 ]
do

echo "1. Siapa yang aktif"
echo "2. Tanggal hari ini"
echo "3. Kalender bulan ini"
echo "4. Keluar"

echo "    Pilihan : \c"

read PILIH

if [ $PILIH -eq 4 ]
then

        break

fi

clear

done

echo "Program berlanjut di sini setelah break"
```

Percobaan 10 : Konstruksi Until

1. Buatlah prog22.sh dengan editor vi, kemudian jalankan

```
$vi prog22.sh

I=1

Until [ $I -gt 10 ]

do
```

```
        echo ""

        let I+=1

done
```

Percobaan 11 : Instruksi dummy

1. Modifikasi file prog21.sh

```
$ vi prog21.sh

#!/bin/sh

# Prog: prog21.sh

PILIH=1

while :
do
echo "1. Siapa yang aktif"
echo "2. Tanggal hari ini"
echo "3. Kalender bulan ini"
echo "4. Keluar"
echo "  Pilihan : \c"
read PILIH
if [ $PILIH -eq 4 ]
then
        break
fi
clear
done

echo "Program berlanjut di sini setelah break"
```

2. Jalankan program prog21.sh

```
$ . prog21.sh
```

3. Buatlah file prog23.sh yang berisi instruksi dummy untuk konstruksi if

```

$ vi prog23.sh

#!/bin/sh

# Prog: prog23.sh

echo -n "Masukkan nilai : "

read A

if [ $A -gt 100 ]

then

    :

else

    echo "OK !"

fi

```

4. Jalankan program prog23.sh beberapa kali dengan input yang berbeda

```
$ . prog23.sh
```

Percobaan 12 : Fungsi

1. Buatlah file fungsi.sh

```

$ vi fungsi.sh

#!/bin/sh

# Prog: fungsi.sh

F1( ) {

    echo "Fungsi F1"

    return 1

}

echo "Menggunakan Fungsi"

F1

F1

echo $?

```

2. Jalankan program fungsi.sh


```
$ . fungsi.sh
```

3. Menggunakan variable pada fungsi dengan memodifikasi file fungsi.sh

```
$ vi fungsi.sh
```

```
#!/bin/sh
```

```
# Prog: fungsi.sh
```

```
F1( )
```

```
{
```

```
    Honor=10000
```

```
    echo "Fungsi F1"
```

```
    return 1
```

```
}
```

```
echo "Menggunakan Fungsi"
```

```
F1
```

```
F1
```

```
echo "Nilai balik adalah $?"
```

```
echo "Honor = $Honor"
```

4. Jalankan program fungsi.sh

```
$ . fungsi.sh
```

5. Menggunakan variable pada fungsi dengan memodifikasi file fungsi.sh

```
$ vi fungsi.sh
```

```
#!/bin/sh
```

```
# Prog: fungsi.sh
```

```
F1( )
```

```
{
```

```
    local Honor=10000
```

```

        echo "Fungsi F1"

        return 1
    }

    echo "Menggunakan Fungsi"

    F1

    F1

    echo "Nilai balik adalah $?"

    echo "Honor = $Honor"

```

6. Jalankan program fungsi.sh

```
$ . fungsi.sh
```

E. TUGAS

1. Buatlah program myprog.sh yang memproses parameter \$1, nilai parameter harus berupa string :

```

start

stop

status

restart

reload

```

Bila buka dari string tersebut, maka berikan pesan eror. Sempurnakan program di bawah ini untuk keperluan tersebut

```

#!/bin/sh

# See how we were called

case "$1" in

start)

    echo "Ini adalah start"

```

```
;;
stop)
echo "Ini adalah stop"
;;
*)
echo $"Usage:$0 {start|stop|restart|reload|status}"
;;
esac
return
```

2. Buat sebuah fungsi pada script confirm.sh yang memberikan konfirmasi jawaban Yes, No atau Continue. Jika jawaban Yes, maka beri nilai balik 0, No = 1 dan Continue = 2.

Modifikasi kerangka program berikut untuk memenuhi permintaan tersebut.

```
#!/bin/sh

# Confirm whether we really want to run this service

confirm() {
    local YES="Y"
    local NO="N"
    local CONT="C"
    while :
    do
        echo -n "(Y)es/(N)o/(C)ontinue? {Y} "
        read answer
        answer=`echo "$answer" | tr '[a-z]' '[A-Z]`
        if [ "$answer" = "" -o "$answer" = $YES ]
        then
            return 0
        elif ...
```

```
        then
            return 2
        elif ...
        then
            return 1
        fi
    done
}
```

Test fungsi diatas dengan program berikut :

```
$ vi testp.sh
. confirm.sh
confirm
if [ $? -eq 0 ]
then
    echo "Jawaban YES OK"
elif [ $? =eq 1 ]
then
    echo "Jawaban NO"
else
    echo "Jawaban CONTINUE"
fi
```

Perhatikan baris pertama, adalah loading dari fungsi confirm yang terdapat di script confirm.sh. Setelah eksekusi script tersebut, maka fungsi confirm dapat digunakan.

FORMAT LAPORAN HASIL PRAKTIKUM

LAPORAN HASIL PRAKTIKUM

Nama :

Nim :

Judul Percobaan :

Hasil Percobaan :

HH. TUJUAN

- Mengetahui variable local dan global di shell
- Mengetahui eksekusi bersyarat
- Mengenal fungsi dan cara memanggilnya
- Mampu membuat user interface dan dialog utility
- Mampu melakukan trap command

II. DASAR TEORI

VARIABLE LOKAL DAN GLOBAL PADA SHELL

Secara umum semua variable adalah lokal. Variable lokal digunakan pada shell yang sama, jika anda me-load atau memanggil shell lainnya (dengan mengetik **/bin/bash** pada prompt) maka shell yang baru tersebut akan mengabaikan variable pada shell sebelumnya.

Global shell didefinisikan : Anda dapat mengkopi variable shell yang lama ke shell yang baru. Untuk membuat suatu variable lokal menjadi global digunakan perintah **export**

sintaks :

```
export variable1, variable2,...variableN
```

EKSEKUSI BERSYARAT

Operatornya adalah && (dibaca AND) dan || (dibaca OR)

sintaks :

```
command1 && command2
```

```
command1 || command2
```

anda dapat menggunakan keduanya

sintaks :

```
command1 && command2 || command3
```

jika command1 sukses dieksekusi maka shell akan menjalankan command2 dan jika command1 tidak sukses maka shell akan mengeksekusi command3.

FUNGSI

Fungsi adalah program yang dapat dipanggil oleh program lainnya dengan menggunakan notasi NamaFungsi(). Fungsi memberikan exit status (\$?) yang dinyatakan dengan **return nr**, atau nilai 0 sebagai default. Membuat fungsi diawali dengan nama fungsi, parameter, kemudian blok program yang dinyatakan dalam { ... }.

Contoh :

```
F1() {  
    .....  
    .....  
    return 1  
}
```

Variabel dapat didefinisikan dalam fungsi sebagai variable local atau global. Hal yang perlu diperhatikan, nama variable yang digunakan dalam sebuah fungsi, jangan sampai bentrok dengan nama variable yang sama di luar fungsi, sehingga tidak terjadi isi variable berubah.

USER INTERFACE DAN DIALOG UTILITY

Program/Shell Script yang bagus harus dapat berinteraksi dengan user. Untuk itu kita dapat menggunakan :

1. Command line argument, misal `./coba.sh 4 3`, dimana 4 dan 3 adalah command line argument bagi script `coba.sh`
2. Statemen `echo` dan `read`, untuk membaca input dari prompt.

Selain itu dapat juga menggunakan dialog utility.

sintaks :

```
dialog --tilte {judul} --backtitle {judul belakang} {Box opsi}
```

dimana Box opsi :

```
--yesno {teks} {tinggi} {lebar}
--msgbox {teks} {tinggi} {lebar}
--infobox {teks} {tinggi} {lebar}
--inputbox {teks} {tinggi} {lebar} [{init}]
--textbox {teks} {tinggi} {lebar}
--menu {teks} {tinggi} {lebar} {menu} {tinggi} {tag 1} item1}...
```

TRAP COMMAND

Berfungsi untuk menjebak suatu command.

sintaks :

```
trap {commands} {signal numberlist}
```

<i>Signal Number</i>	<i>Keterangan</i>
0	shell exit
1	hangup
2	interrupt (CTRL+C)

<i>Signal Number</i>	<i>Keterangan</i>
3	quit
9	kill

JJ. LANGKAH – LANGKAH

35. Masuk ke sistem operasi Linux. Login : **mahasiswa** password : **mahasiswa**
36. Lakukan operasi pembuatan variable local dan global.
37. Lakukan eksekusi bersyarat
38. Buatlah fungsi
39. Buat user interface : message box, input box, yesno box, dialog dengan menu
40. Lakukan trap command

KK. PERCOBAAN

Percobaan 1 : Variable Local

1. Ketikkan variable berikut :

```
$vehc=Bus
```

```
$echo $vehc
```
2. Buka shell baru (yang kedua), lihat isi variable

```
$/bin/bash
```

```
$echo $vehc
```
3. Ketik variable di shell yang baru, tampilkan variable tersebut

```
$vehc=Sedan
```

```
$echo $vehc
```
4. Keluar dari shell baru, tampilkan variable

```
$exit
```

```
$echo $vehc
```

Percobaan 2: Variable Global

1. Ketik variable berikut :

```
$vehc=Bus
```

```
$echo $vehc
```

2. Export variable tersebut agar menjadi global, masuk ke shell baru, lihat hasilnya

```
$export vehc
```

```
$/bin/bash
```

```
$echo $vehc
```

3. Keluar dari shell baru

```
$exit
```

```
$echo $vehc
```

Percobaan 3 : Eksekusi Bersyarat

1. Ketik command berikut :

```
$rm myfile && echo "File berhasil dihapus" || echo "File  
tidak terhapus"
```

Percobaan 4 : Fungsi

1. Ketik fungsi berikut setelah prompt

```
$ Hallo()  
  
{  
  
echo "Hallo $LOGNAME, have a nice day"  
  
return  
  
}
```

2. Panggil fungsi tersebut

```
$Hallo
```

3. Fungsi ini akan hilang ketika komputer di-restart. Untuk mengatasi hal ini dan untuk menambahkan fungsi anda dan untuk otomatisasi task harian, masukkan fungsi ke dalam file **/etc/bashrc**. Untuk ini anda harus logon sebagai root.

```
$ su -l
```

4. Buka file **/etc/bashrc**

```
#vi /etc/bashrc
```

4. Tekan shift+G untuk ke akhir file, tambahkan fungsi **today()** berikut untuk mencetak format tanggal :

```
# today() untuk mencetak tanggal
```

```
# untuk menjalankan fungsi ini ketik today setelah prompt
$ atau #

today()

{
    echo Hari ini adalah `date +%A %d in %B of %Y
(%r)` `
    return
}

```

5. Simpan file, lihat file setelah dimodifikasi

```
#cat /etc/bashrc
```

6. Fungsi ini akan tersedia untuk semua user. Untuk menjalankan fungsi ini anda harus exit dahulu, lalu login kembali, kemudian jalankan fungsi ini

```
#today
```

```
$today
```

7. Untuk membuat fungsi bagi user tertentu dan hanya tersedia bagi user tertentu maka buka file .bashrc di home directory user

```
$ vi .bashrc
```

8. Tekan (Shift+G) untuk ke akhir file, kemudian ketik contoh fungsi berikut

```
Hai()

{
    echo "Hai $LOGNAME ! Tidak akan ada perubahan,
kecuali anda login kembali !"
    echo "Tekan sembarang tombol..."
    read
    return
}

```

9. Simpan lalu exit. Untuk menjalankan fungsi ini anda harus exit dahulu lalu login kembali. Fungsi ini akan tersedia hanya bagi user anda, tidak bagi semua user di sistem.

```
$ Hai
```

Note : gunakan file .bashrc di home direktori anda untuk menambahkan alias dan fungsi

Percobaan 6 : file .bash_logout

1. Untuk membuat pesan atau aksi ketika anda logout, buka file .bash_logout lalu tambahkan apa yang ingin anda jalankan ketika logout. Misal ketika logout anda ingin menampilkan pesan goodbye

```
$vi .bash_logout
```

2. Lalu ketik pesan

```
echo "Goodbye $LOGNAME...tekan sembarang tombol"
```

Percobaan 7 : User Interface

1. Buat script di bawah ini untuk membaca masukan dari user

```
$ cat > userinterf

# script menggunakan echo dan read untuk interaksi user

echo "Ketikkan nama anda..please :"

read nama

echo "Umur anda..please :"

read umur

umurbaru=`expr $umur + 1`

echo "hallo $nama, tahun depan umurmu $umurbaru.."
```

2. Simpan lalu jalankan

```
$chmod 755 userinterf

$./userinterf
```

Percobaan 8 : User Interface

1. Buat script berikut :

```
$cat > menu1

while :
```

```

do

    clear

    echo "-----"

    echo "          Menu Utama"

    echo "-----"

    echo "[1] Lihat tanggal hari ini"
    echo "[2] Lihat file pada direktori"
    echo "[3] Lihat kalender"
    echo "[4] Buka teks editor untuk mengetik surat"
    echo "[5] Keluar"

    echo "===== "

    echo -n "Pilihan anda [1-5]: "

    read pil

    case $pil in

        1) echo "Hari ini `date`, tekan enter..." ; read ;;

        2) echo "File di direktori `pwd`" ; ls -l ; echo
        "Tekan enter..." ; read ;;

        3) cal ; echo "Tekan enter..." ; read ;;

        4) vi ;;

        5) exit 0 ;;

        *) echo "Ooopss..silakan pilih 1-5..";
           echo "Tekan enter.." ; read ;;

    esac

done

```

2. Jalankan script tersebut

Percobaan 9 : Dialog Utility

1. Ketik script berikut sebagai contoh penggunaan dialog utility.
\$cat > dialog1

```
dialog --title "Linux Dialog Utility Infobox" --backtitle
"Belajar Shell Script Linux" --infobox "Dialog Box ini disebut
infobox, untuk menampilkan informasi di layar..Tekan sembarang
tombol..." 7 50 ; read
```

2. Simpan dan jalankan

```
$chmod +x dialog1
```

```
$/dialog1
```

Percobaan 10 : Message Box

1. Ketik script ini untuk membuat message box

```
$cat > dialog2
```

```
dialog --title "Linux Dialog Utility Message Box" --
backtitle "Belajar Shell Script Linux" --msgbox "Dialog Box
ini disebut message box, untuk menampilkan informasi di layar
beserta tombol OK..Tekan sembarang tombol..." 9 50 ;
```

2. Simpan dan jalankan

```
$chmod +x dialog2
```

```
$/dialog2
```

Percobaan 11 : YesNo Box

1. Ketik script berikut untuk membuat yesno box

```
$cat > dialog3
```

```
dialog --title "Peringatan : Penghapusan File" --
backtitle "Belajar Shell Script Linux" --yesno "\nYakin ingin
menghapus '/usr/surat/lamarankerja' file" 7 60
```

```
sel=$?
```

```
case $sel in
```

```
0)echo "user akan menghapus file";;
```

```
1)echo "user tidak jadi menghapus file";;
```

```
255)echo "dibatalkan user, dengan menekan tombol
[ESC]";;
```

```
esac
```

2. Simpan dan jalankan

```
$chmod +x dialog3
```

```
$/dialog3
```

Percobaan 12 : InputBox

1. Ketik script berikut untuk membuat input box

```
$cat > dialog4

dialog --title "Inputbox - mengambil input dari user" --
backtitle "Belajar Shell Script Linux" --inputbox "Silakan
masukkan nama anda" 8 60 2>/tmp/input.$$

sel=$?

na=`cat /tmp/input.$$`

case $sel in

    1)echo "Hallo $na" ;;
    2)echo "Cancel ditekan" ;;
    255) echo "Tombol [ESC] ditekan" ;;

esac

rm -f /tmp/input.$$
```

2. Simpan dan jalankan

```
$chmod +x dialog4

$/dialog4
```

Percobaan 13 : Dialog Menu

1. Ketik script berikut untuk membuat menu dengan dialog

```
$cat > dialogmenu

dialog --backtitle "Belajar Shell Script Linux" --title
"Menu Utama" --menu "Gunakan panah [UP] [DOWN], [ENTER] untuk
memilih" 15 50 3 Tanggal "Menampilkan Tanggal" Kalender
"Melihat Kalender" Editor "Membuka Editor Vi "
2>/tmp/menuitem.$$

menuitem=`cat /tmp/menuitem.$$`

opt=$?

case $menuitem in

    Tanggal) date;;
```

```

        Kalender) cal;;

        Editor) vi;;

    esac

```

2. Simpan dan jalankan

```

$rm -f /tmp/menuitem.$$

$chmod +x dialogmenu

$./dialogmenu

```

Percobaan 14 : Trap Command

1. Ketik command berikut

```

$cat > tessinyal

ls -R /

```

2. Simpan dan jalankan

```

$chmod +x tessinyal

$./tessinyal

```

3. Tekan [ctrl]+c untuk menghentikannya. Ctrl+c merupakan sinyal

4. Sekarang buat script di bawah ini

```

$cat > tessinyal1

#Mengapa harus menjebak sinyal ?

Take_input()

{

    recno=0

    clear

    echo "Aplikasi Catatan Appointment di Linux"

    echo -n "Masukkan nama file database : "

    read filename

    if [ ! -f $filename ];then

```



```

        echo "Maaf, $filename tidak ada, sekarang akan
dibuat $filename database"

        echo "Aplikasi Catatan Appointment database
file" > $filename

    fi

    echo "Pemasukan data pada tanggal: `date`"
>/tmp/input0.$$

while :
do

    echo -n "Judul Appointment : "
    read nama

    echo -n "Time : "
    read waktu

    echo -n "Catatan : "
    read catatan

    echo -n "Data Okay (y/n) ?"
    read jawaban

    if [ $jawaban = y -o $jawaban = Y ]; then
        recno=`expr $recno + 1`
        echo "$recno. $nama $waktu $catatan" >>
/tmp/input0.$$
    fi

    echo "Tambah data lagi (y/n) ?"
    read isnext

    if [ $isnext = n -o $isnext = N ];then
        cat /tmp/input0.$$ >> $filename
        rm -f /tmp/input0.$$
    fi
done

```

```

        return #stop loop
    fi
done

}

#panggil fungsi
Take_input

```

5. Simpan dan jalankan

```

$chmod +x tessinyal

$./tessinyal

```

6. Setelah mengisi beberapa data, tampilkan isi database dengan perintah cat

```

$cat namadatabase

```

7. Jalankan lagi file tersebut

```

$./tessinyal

```

Cukup isi satu data saja, setelah itu tekan ctrl+c sebelum menjawab pertanyaan “Tambah data (y/n)” maka program akan terminated(berhenti), dan temporary file akan tertinggal di /tmp direktori.

8. Cek keberadaan temporary filename

```

$ls /tmp/input*

```

9. Untuk menghindari hal di atas maka kita perlu mendeteksi sinyal dengan menggunakan trap command. Untuk itu tambahkan trap statement sebelum memanggil fungsi Take_input, yaitu del_file 2. Trap statementnya yaitu fungsi del_file, 2 berarti penekanan CTRL+C. Buka file script di atas dan modifikasi sehingga menjadi sebagai berikut :

```

$vi tessinyal

del_file()

{

    echo" Penjebakan penekanan tombol [ctrl]+c"

    rm -f /tmp/input0.$$

    exit 1

}

Take_input()

{

```

```

recno=0

clear

echo "Aplikasi Peningat Appointment di Linux"

echo -n "Masukkan nama file database : "

read filename

if [ ! -f $filename ];then

    echo "Maaf, $filename tidak ada, sekarang akan
dibuat $filename database"

    echo "Aplikasi Peningat Appointment database
file" > $filename

fi

echo "Pemasukan data pada tanggal: `date`"
>/tmp/input0.$$

while :
do

    echo -n "Judul Appointment :"
    read nama

    echo -n "Waktu :"
    read waktu

    echo -n "Catatan :"
    read catatan

    echo -n "Data Okay (y/n) ?"
    read jawaban

    if [ $jawaban = y -o $jawaban = Y ]; then
        recno=`expr $recno + 1`
        echo "$recno. $nama $waktu $catatan" >>
/tmp/input0.$$

```

```

        fi

        echo "Tambah data lagi (y/n) ?"

        read isnext

        if [ $isnext = n -o $isnext = N ];then

        cat /tmp/input0.$$ >> $filename

        rm -f /tmp/input0.$$

        return #stop loop

        fi

    done

}

#memanggil fungsi

trap del_file 2

Take_input

```

10. Jalankan script di atas
\$./tessinyall
11. Coba tekan CTRL+C ketika pengisian data. Periksa temporary file
\$ls /tmp/input*

FORMAT LAPORAN HASIL PRAKTIKUM

LAPORAN HASIL PRAKTIKUM

Nama :

Nim :

Judul Percobaan :

Hasil Percobaan :