MODUL PRAKTIKUM
SISTEM OPERASI

Tim Dosen Teknik Informatika
dan Sistem Informasi

FAKULTAS ILMU KOMPUTER
UNIVERSITAS ESA UNGGUL

PERTEMUAN 1

Perintah Dasar OS Linux

Praktikum 1

Perintah Dasar Sistem Operasi Linux

POKOK BAHASAN:

- ✓ Format Instruksi pada Sistem Operasi Linux
- ✓ Perintah-Perintah Dasar pda Sistem Operasi Linux

TUJUAN BELAJAR:

Setelah mempelajari materi dalam bab ini, mahasiswa diharapkan mampu:

- ✓ Menggunakan perintah-perintah dasar untuk informasi user
- ✓ Mengenal format instruksi pada system operasi Linux
- ✓ Menggunakan perintah-perintah dasar pada system operasi Linux
- ✓ Menggunakan utilitas dasar pada system operasi Linux

DASAR TEORI:

Setiap pemakai LINUX harus mempunyai nama login (user account) yang sebelumnya harus didaftarkan pada administrator system. Nama login umumnya dibatasi maksimum 8 karakter dan umumnya dalam huruf kecil. Prompt dari shell bash pada LINUX menggunakan tanda "**$**".

Sebuah sesi LINUX terdiri dari :

1. Login
2. Bekerja dengan Shell / menjalankan aplikasi
3. Logout

Tergantung atas shell yang digunakan, pada Linux bash maka pada proses login akan mengeksekusi program */etc/profile* (untuk semua pemakai) dan file *.base_profile* di direktori awal (HOME) masing- masing.

Pada saat logout, maka program shell bash akan mengeksekusi script yang bernama *.bash_logout*.

1 FORMAT INSTRUKSI LINUX

Instruksi Linux standar mempunyai format sebagai berikut :

```
$ NamaInstruksi [pilihan] [argumen]
```

Pilihan adalah option yang dimulai dengan tanda – (minus). Argumen dapat kosong, satu atau beberapa argumen (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

2 MANUAL

Linux menyediakan manual secara on-line. Beberapa kunci keyboard yang penting dalam menggunakan manual adalah :

`Q`	untuk keluar dari program man
`<Enter>`	ke bawah, baris per baris
`<Spasi>`	ke bawah, per halaman
`b`	kembali ke atas, 1 halaman
`/teks`	mencari teks (string)
`n`	meneruskan pencarian string sebelumnya

Manual dibagi atas Bab-bab sebagai berikut :

Bab	Isi
1	User commands
2	System calls
3	Library calls
4	Devices
5	File formats
6	Games
7	Miscellaneous
8	System commands
9	Kernel internals
N	Tcl/Tk command

TUGAS PENDAHULUAN :

Jawablah pertanyaan-pertanyaan di bawah ini :

1. Apa yang dimaksud perintah informasi user di bawah ini :
   `id, hostname, uname, w, who, whoami, chfn, finger`
2. Apa yang dimaksud perintah dasar di bawah ini :
   `date, cal, man, clear, apropos, whatis`
3. Apa yang dimaksud perintah-perintah manipulasi file di bawah ini : `ls, file, cat, more, pg, cp, mv, rm, grep`

PERCOBAAN :

1. Login sebagai user.
2. Bukalah Console Terminal dan lakukan percobaan-percobaan di bawah ini
3. Selesaikan soal-soal latihan

Percobaan 1 : Melihat identitas diri (nomor id dan group id)

```
$ id
```

Percobaan 2 : Melihat tanggal dan kalender dari sistem

1. Melihat tanggal saat ini `$ date`

2. Melihat kalender

```
$ cal 9 2002
$ cal -y
```

Perc obaan 3 : Melihat identitas mesin

```
$ hostname
$ uname
$ uname -a
```

Percobaan 4 : Melihat siapa yang sedang aktif

1. Mengetahui siapa saja yang sedang aktif

```
$ w $
who
$ whoami
```

2. Mengubah informasi finger

```
$ chfn <user>
Changing finger information for
student. Password:
Name[user wks]: <Nama Pengguna di
wks> Office[]: Lab Pemrograman 2
Office Phone []: 2301
Home Phone []: 5947280

Finger information changed.
```

3. Melihat informasi finger

```
$ finger
$ finger <user>
```

Percobaan 5 : Menggunakan manual

```
$ man ls
$ man man
$ man –k file
$ man 5 passwd
```

Percobaan 6 : Menghapus layar

```
$ clear
```

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

```
$ apropos date
$ apropos mail
$ apropos telnet
```

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

```
$ whatis date
```

Percobaan 9 : Manipulasi berkas (file) dan direktori

1. Menampilkan current working director y `$ 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
```

Percobaan 10 : Melihat tipe file

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

Percobaan 11 : 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
```

Percobaan 12 : Melihat isi file

1. Menggunakan instruksi cat
```
$ cat f1
```
2. Menampilkan file per satu layar penuh
```
$ more f1
$ pg f1
```

Percobaan 13 : 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 mdir
```

Percobaan 14 : Menghapus file

```
$ rm f1
$ cp mydir/f1 f1
$ cp mydir/f2 f2
$ rm f1
$ rm –i f2
```

Percobaan 15 : Mencari kata atau kalimat dalam file

```
$ grep root /etc/passwd
$ grep “:0:” /etc/passwd
$ grep student /etc/passwd
```

LATIHAN:

1. Ubahlah informasi finger pada komputer Anda.
2. Lihatlah user-user yang sedang aktif pada komputer Anda.
3. Perintah apa yang digunakan untuk melihat kalender satu tahun penuh ?
4. Bagaimana anda dapat melihat manual dari perintah `cal` ?

5. Bagaimana melihat perintah manual `ls` dengan kata kunci sort ?
6. Bagaimana tampilan untuk perintah `ls -a -l` dan `ls -al` ?
7. Tampilkan semua file termasuk yang hidden file pada direktori `/etc`.
8. Tampilkan semua file secara lengkap pada direktori `/etc`.
9. Buatlah direktori `prak1` pada direktori aktif, kemudian copy-kan file `/etc/group` ke file `tes1`, `tes2` dan `tes3` pada direktori ini.
10. Tampilkan isi file `tes1` per satu layar penuh.
11. Pindahkan file `tes1` dan `tes2` ke home direktori.
12. Hapus file `tes1` dan `tes` dengan konfirmasi.

LAPORAN RESMI:

1. Buatlah summary Percobaan 1 sampai dengan percobaan 15 dalam bentuk table seperti di bawah ini :

Perintah	Deskripsi	Format
`id`		
`date`		
`cal`		
`hostname`		
`uname`		
`w`		
`who`		
`whoami` `chfn`		

2. Analisa latihan yang telah dilakukan.
3. Berikan kesimpulan dari praktikum ini.

MODUL PRAKTIKUM

SISTEM OPERASI

PERTEMUAN 2

Operasi Input Output

Universitas Esa Unggul

Jakarta – 2008

Praktikum 2

Operasi Input Output

POKOK BAHASAN:

- ✓ Pipeline
- ✓ Redirection

TUJUAN BELAJAR:

Setelah mempelajari materi dalam bab ini, mahasiswa diharapkan mampu:

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

DASAR TEORI:

1 PROSES I/O

Sebuah proses memerlukan Input dan Output.

Instruksi (*command*) yang diberikan pada Linux melalui Shell disebut sebagai *eksekusi program* yang sela njutnya 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/o utput adalah :

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

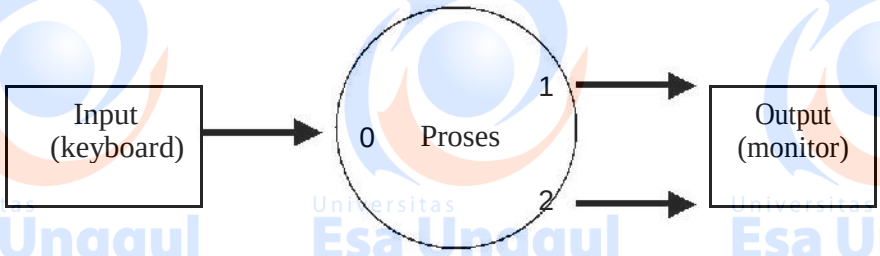
2 FILE DESCRIPTOR

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)

4. 1 = layar (standar output)
5. 2 = layar (standar error)



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

3 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 inp ut

1> atau > pengganti standard output

2>

4 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 menghubngkan Proses 1 dengan Proses2 dan dinyatakan dengan symbol "|".

Proses1 | Proses2

5 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 ……. | P_{n-1} | P_n

Maka P_2 sampai dengan P_{n-1} mutlak harus utilitas Linux yang berfungsi sebagai filter. P_1 (awal) dan P_n (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 :

4. Perintah `grep`

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

5. 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.

4. Perintah `sort`

   Digunakan untuk mengurutkan masukannya berdasarkan urutan nomor ASCII dari karakter.

5. Perintah `cut`

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

6. Perintah `uniq`

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

TUGAS PENDAHULUAN :

Jawablah pertanyaan-pertanyaan di bawah ini :

3. Apa yang dimaksud *redirection* ?
4. Apa yang dimaksud *pipeline* ?
5. Apa yang dimaksud perintah di bawah ini :

   `echo, cat, more, sort, grep, wc, cut, uniq`

PERCOBAAN :

2. Login sebagai user.
3. Bukalah Console Terminal dan lakukan percobaan-percobaan di bawah ini. Perhatikan hasil setiap percobaan.
4. Selesaikan soal-soal latihan.

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 ^d
exit dengan ^d
[Ctrl-d]
```

3. Input dari keyboard dan output ke alamat internet

```
$ mailarna@eepis-
its.educontoh surat yang
langsung
dibuat pada standard input
(keyboard) [Ctrl-d]
```

4. 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
```

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

```
$ ls filebaru        (Terdapat pesan error)
$ ls filebaru 2> out.txt
$ cat out.txt
$ ls filebaru 2> out.txt 2>&1
$ cat out.txt
```

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

```
$ echo "mencoba menulis file" 1>
baru $ cat filebaru 2> baru 1>&2
$ cat baru
```

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!
%%%
```

8. Notasi – (input keyboard) adalah representan input dari keyboard. Artinya menampilkan file 1, kemudian menampilkan input dari keyboard dan menampilkan file 2. Perhatikan bahwa notasi "-" berarti menyelipkan input dari keyboard

```
$ cat myfile.txt – surat
```

9. Untuk membelokkan standart output ke file, digunakan operator >

```
$ echo hello
$ echo hello > output
$ cat output
```

10. Untuk menambahkan output ke file digunakan operator >>

```
$ echo bye >>
output $ cat output
```

11. Untuk membelokkan standart input digunakan operator

```
<$ cat < output
```

12. Pembelokan standart input dan standart output dapat dikombinasikan tetapi tidak boleh menggunakan nama file yang sama sebagai standart input dan output.

```
$ cat < output >
out $ cat out
$ cat < output >>
out $ cat out
$ cat < output >
output $ cat output
$ cat < out >> out          (Proses tidak berhenti)
[Ctrl-c]
$ cat out
```

Percobaan 3 : Pipa (*pipeline*)

1. Operator pipa (|) digunakan untuk membuat eksekusi proses dengan melewati data langsung ke data lainnya.

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

Percobaan 4 : Filter

2. Pipa juga digunakan untuk mengkombinasikan utilitas sistem untuk membentuk fungsi yang lebih kompleks

```
$ w –h | grep <user>
$ grep <user> /etc/passwd
$ ls /etc | wc
$ ls /etc | wc –l
$       cat       >
kelas1.txtBadu
Zulkifli
Yulizir
Yudi Ade
[Ctrl-d]

$ cat >
kelas2.txtBudi
Gama
Asep
Muchlis
[Ctrl-d]
$ cat kelas1.txt kelas2.txt | sort
$ cat kelas1.txt kelas2.txt > kelas.txt
$ cat kelas.txt | sort | uniq
```

LATIHAN:

1. Lihat daftar secara lengkap pada direktori aktif, belokkan tampilan standard output ke file `baru`.
2. Lihat daftar secara lengkap pada direktori `/etc/passwd`, belokkan tampilan standard output ke file `baru` tanpa menghapus file `baru` sebelumnya.
3. Urutkan file `baru` dengan cara membelokkan standard inp ut.
4. Urutkan file `baru` dengan cara membelokkan standard input dan standard output ke file `baru.urut`.
5. Buatlah direktori `latihan2` 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 >
hello.txtdog cat
cat duck
dog chicken
chicken duck
chicken cat
dog duck
[Ctrl-d]
$ cat hello.txt | sort | uniq
$ cat hello.txt | grep “dog” | grep –v “cat”
```

LAPORAN RESMI:

1. Analisa hasil percobaan 1 sampai dengan 4, untuk setiap perintah jelaskan tampilannya.
2. Kerjakan latihan diatas dan analisa hasilnya
3. Berikan kesimpulan dari praktikum ini.

MODUL PRAKTIKUM

SISTEM OPERASI

PERTEMUAN 3

Operasi File & Struktur Direktori

Universitas Esa Unggul

Jakarta – 2008

Praktikum 3

Operasi File dan Struktur Direktory

POKOK BAHASAN:

- ✓ Operasi File pada Sistem Operasi Linux
- ✓ Struktur Direktory pada Sistem Operasi Linux

TUJUAN BELAJAR:

Setelah mempelajari materi dalam bab ini, mahasiswa diharapkan mampu:

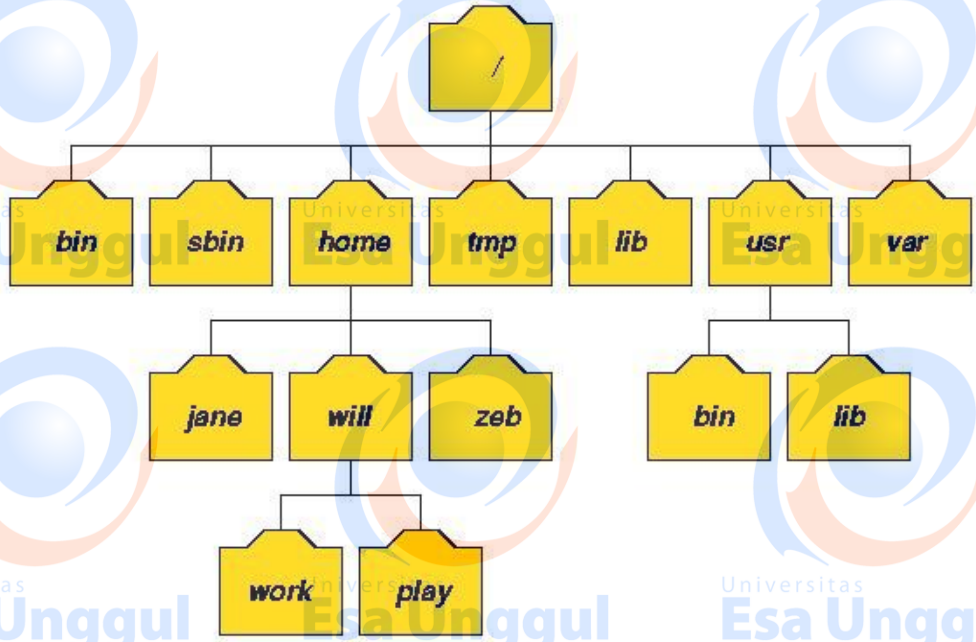
- ✓ Memahami organisasi file dan direktory pada sistem operasi Linux
- ✓ Menciptakan dan manipulasi directory
- ✓ Memahami konsep link dan symbolic link

DASAR TEORI:

1 ORGANISASI FILE

Sistem file pada Linux menyerupai pepohonan (tree), yaitu dimulai dari root, kemudian direktori dan sub dirrektori. Sistem file pada Linux diatur secara hirarkhikal, yaitu dimulai dari root dengan symbol "/" seperti Gambar 3.1.

Kita dapat menciptakan File dan Direktori mulai dari root ke bawah. Direktori adalah file khusus, yang berisi nama file dan INODE (pointer yang menunjuk ke data / isi file tersebut). Secara logika, Direktori dapat berisi File dan Direktori lagi (disebut juga Subdirektori).



Gambar 1.3 Struktur direktori pada Linux

2 DIREKTORY STANDAR

Setelah proses instalasi, Linux menciptakan system file yang baku, terdiri atas direktori sebagai berikut :

Direktori	Deskripsi
`/etc`	Berisi file administrative (konfigrasi dll) dan file executable atau script yang berguna untuk administrasi system.
`/dev`	Berisi file khusus yang merepresentasikan peralatan hardware seperti memori, disk, printer, tape, floppy, jaringan dll.
`/bin`	Berisi utilitas sistem level rendah (binary) .
`/sbin`	Berisi utilitas sistem untuk superuser (untuk membentuk administrasi sistem).
`/usr/sbin` `/usr/bin`	Berisi utilitas sistem dan program aplikasi level tinggi.
`/usr/lib`	Berisi program library yang diperlukan untuk kompilasi

	program (misalnya C). Berisi instruksi (command) misalnya untuk Print Spooler (lpadmin) dll.
/tmp	Berisi file sementara, yang pada saat Bootstrap akan dihapus (dapat digunakan oleh sembarang user).
/boot	Berisi file yang sangat penting untuk proses bootstrap. Kernel **vmlinuz** disimpan di direktori ini.
/proc	Berisi informasi tentang kernel Linux, proses dan virtual system file.
/var	Direktori variable, artinya tempan penyimpanan LOG (catatan hasil output program), file ini dapat membengkak dan perlu dimonitor perkembangannya.
/home /mnt	Berisi direktori untuk pemakai Linux (pada SCO diletakkan pada /usr) Direktori untuk mounting system file
/root	Home direktori untuk superuser (root)
/usr/bin/X11	Symbolic link ke /usr/X11R6/bin, program untuk X-Window
/usr/src	Source code untuk Linux
/opt	Option, direktori ini biasanya berisi aplikasi tambahan ("add-on") seperti Netscape Navigator, kde, gnome, applix dll.

Direktori /etc

Berisi file yang berhubungan dengan administrasi system, maintenance script, konfigurasi, security dll. Hanya superuser yang boleh memodifikasi file yang berada di drektori ini. Subdirektori yang sering diakses pada direktori **/etc** antara lain :

6. `httpd`, apache web server.
7. `ppp`, point to point protocol untuk koneksi ke Internet.
8. `rc.d` atau `init.d`, inisialisasi (startup) dan terminasi (shutdown) proses di Linuxdengan konsep runlevel.
9. `cron.d`, rincian proses yang dieksekusi dengan menggunakan jadwal(timedependent process)

6. `FILES`, file security dan konfigurasi meliputi :*passwd, hosts, shadow, ftpaccess,inetd.conf, lilo.conf, motd, printcap, profile, resolv.conf, sendmail.cf, syslog.conf, dhcp.conf, smb.conf, fstab* .

Direktori `/dev`

Konsep Unix dan Linux adalah memperlakukan peralatan hardware sama seperti penanganan file. Setiap alat mempunyai nama file yang disimpan pada direktori `/dev`.

Peralatan	Direktori
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`
Parallel Port	LPT1 : `/dev/lp0` LPT2 : `/dev/lp1`
Serial Port	COM1 : `/dev/ttyS0` COM2 : `/dev/ttyS1` Universal : `/dev/modem` (link dari S0 atau S1)

Direktori `/proc`

Direktori `/proc` adalah direktori yang dibuat diatas RAM (Random Access Memory) dengan system file yang diatur oleh kernel. `/proc` berisi nomor proses dari system dan nama driver yang aktif di system. Semua direktori berukuran 0 (kosong) kecuali file `kcore` dan `self`. Setiap nomor yang ada pada direktori tsb merepresentasikan `PID` (Process ID).

3 TIPE FILE

Pada Linux terdapat 6 buah tipe file yaitu

7. Ordinary file
8. Direktori
9. Block Device (Peralatan I/O)

   Merupakan representasi dari peralatan hardware yang menggunakan transmisi data per block (misalnya 1 KB block), seperti disk, floppy, tape.
10. Character Device (Peralatan I/O)

    Merupakan representasi dari peralatan hardware yang menggunakan transmisi data karakter per karakter, seperti terminal, modem, plotter dll
11. Named Pipe (FIFO)

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

4 PROPERTI FILE

File mempunyai beberapa atribut, antara lain :

- Tipe file : menentukan tipe dari file, yaitu :

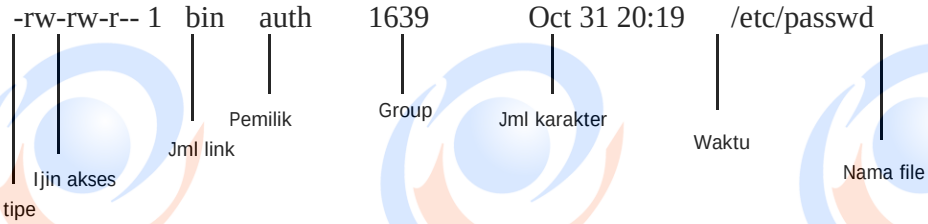
Karakter	Arti
-	File biasa
d	Direktori
l	Symbolic link
b	Block special file
c	Character special file
s	Socket link
p	FIFO

- 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

6. Waktu pembuatan : menentukan kapan file terakhir dimodifikasi
- Nama file : menentukan nama file yang dimaksud

Contoh :



5 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
```

6 SIMBOLIC 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
```

`fileDuplikat` disebut *hard link* dimana kedua file akan muncul identik (*linkcount* = 2) Bila `fileAsli` atau `ileDuplikat` diubah perubahan akan terjadi pada file lainnya.

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
```

Pilihan `–s` (*shortcut*) merupakan bentuk *soft link* dimana jumlah *link count* pada file asal tidak akan berubah. Pada bentuk *soft link, symbolic link* dapat dilakukan pada file yang tidak ada, sedangkan pada *hard link* tidak dimungkinkan. Perbedaan lain, *symbolic link* dapat dibentuk melalui media disk atau partisi yang berbeda dengan *soft link*, tetapi pada *hard link* terbatas pada partisi disk yang sama.

7 MELIHAT ISI FILE

Untuk melihat jenis file menggunakan format :

```
file filename(s)
```

Isi file akan dilaporkan dengan deskripsi level tinggin seperti contoh berikut

```
$ file myprog.c letter.txt webpage.html
myprog.c:       C program text
letter.txt:     ASCII text
webpage.html:   HTML document text
```

Perintah ini dapat digunakan secara luas untuk file yang kadang membingungkan, misalnya antara kode C++ dan Java.

8 MENCARI FILE

Jika ingin melihat bagaimana pohon direktori dapat digunakan perintah

5. find

   Format : `find`*`directory`*`–name`*`targetfile`*`-print`

   Akan melihat file yang bernama *`targetfile`* (bisa berupa karakter wildcard)

6. which

   Format : `which`*`command`*

Untuk mengetahui letak system utility

3. locate

   Format : `locate`*`string`*

   Akan me ncari file pada semua directori dengan lebih cepat dan ditampilkan dengan path yang penuh.

9 MENCARI TEXT PADA FILE

Untuk mencari text pada file digunakan perintah grep (*General RegularExpression Print*) dengan format perintah

`grep` *`option pattern files`*

Grep akan mencari file yang bernama sesuai pattern yang diberikan dan akan menampilkan baris yang sesuai.

TUGAS PENDAHULUAN :

Jawablah pertanyaan-pertanyaan di bawah ini :

1. Apa yang dimaksud perintah-perintah direktory : `pwd, cd, mkdir, rmdir`.
5. Apa yang dimaksud perintah-perintah manipulasi file : `cp, mv` dan `rm` (sertakan format yang digunakan)
6. Jelaskan perbedaan *Symbolic link* menggunakan *hard link* (*direct*) dan *soft link* (*indirect*).
7. Tuliskan maksud perintah-perintah : `file, find, which, locate` dan `grep`.

PERCOBAAN :

9. Login sebagai user.
10. Bukalah Console Terminal dan lakukan percobaan-percobaan di bawah ini. Perhatikan hasilnya.
11. Selesaikan soal-soal latihan

Percobaan 1 : Direktory

1. Melihat direktori HOME

```
$ pwd
$ echo $HOME
```

2. Melihat direktori aktual dan parent direktori

```
$ pwd
$ cd .
$ pwd
$ cd ..
$ pwd
$ cd
```

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. Menghap us 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, mengapa ?)
$ ls -l B
$ rmdir B/F B
$ ls -l B          (Terdapat pesan error, me ngapa ?)
```

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

```
$ pwd
$ ls -l
$  cd  A
$ pwd
$ cd ..
$ pwd
$ cd /home/<user>/C
$ pwd
$ cd /<user/C(Terdapat pesan error, mengapa ?)  $
pwd
```

Percobaan 2 : Manipulasi file

1. Perintah cp untuk mengkopi file atau seluruh direktori

```
$ cat >
contohMembuat
sebuah file [Ctrl-
d]
$ cp contoh
contoh1 $ ls -l
$ cp contoh A
$ ls –l A
$ cp contoh contoh1
A/D $ ls –l A/D
```

2. Perintah mv untuk memindah file

```
$ mv contoh
contoh2 $ ls -l
$ mv contoh1 contoh2
A/D $ ls –l A/D
$ mv contoh contoh1
C $ ls –l C
```

3. Perintah rm untuk menghapus file

```
$ rm contoh2
$ ls -l
$ rm –i contoh
$ rm –rf A C $
ls -l
```

Percobaan 3 : *Symbolic Link*

13. Membuat shortcut (file link)

```
$ 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
$ ln -s z bye.txt
$ ls -l bye.txt
$ cat bye.txt
```

Percobaan 4 : *Melihat Isi File*

```
$ ls –l
$ file halo.txt
$ file bye.txt
```

Percobaan 5 : *Mencari file*

1. Perintah find

```
$ find /home –name “*.txt” –print >
myerror.txt $ cat myerror.txt
$ find . –name “*.txt” –exec wc –l ‘{}’ ‘;’
```

3. Perintah which

```
$ which ls
```

4. Perintah locate

```
$ locate “*.txt”
```

Percobaan 6 : *Mencari text pada file*

```
$ grep Hallo *.txt
```

LATIHAN:

1. Cobalah urutan perintah berikut :

```
$ cd
$ pwd
$ ls –al
$ cd . $
pwd
$ cd ..
$ pwd
$ ls -al
$  cd ..
$ pwd
$ ls -al
$ cd /etc
$ ls –al | more
$ cat passwd
$ cd –
$ pwd
```

2. Lanjutkan penelusuran pohon pada sistem file menggunakan `cd`, `ls`, `pwd` dan `cat`. Telusuri direktory `/bin, /usr/bin, /sbin, /tmp` dan `/boot`.
7. Telusuri direktory `/dev`. Identifikasi perangkat yang tersedia. Identifikasi tty (termninal) Anda (ketik `who am i`); siapa pemilih tty Anda (gunakan `ls -l`).
9. Telusuri derectory `/proc`. Tampilkan isi file `interrupts, devices,cpuinfo, meminfo` dan `uptime` menggunakan perintah `cat`. Dapatkah Anda

   melihat mengapa directory `/proc` disebut *pseudo -filesystem* yang memungkinkan akses ke struktur data kernel ?
10. Ubahlah direktory home ke user lain secara langsung menggunakan `cd ~username`.
11. Ubah kembali ke direktory home Anda.
12. Buat subdirektory `work` dan `play`.
13. Hapus subdirektory `work`.
14. Copy file `/etc/passwd` ke direktory home Anda.
15. Pindahkan ke subirectory `play`.
16. Ubahlah ke subdirektory `play` dan buat symbolic link dengan nama `terminal` yang menunjuk ke perangkat tty. Apa yang terjadi jika melakukan *hard link* ke perangkat tty ?
17. Buatlah file bernama `hello.txt` yang berisi kata ”`hello word`”. Dapatkah Anda gunakan ”`cp`” menggunakan ”terminal” sebagai file asal untuk menghasilkan efek yang sama ?
18. Copy `hello.txt` ke terminal. Apa yang terjadi ?
19. Masih direktory home, copy keseluruhan direktory play ke direktory bernama `work` menggunakan symbolic link.
20. Hapus direktory work dan isinya dengan satu perintah

LAPORAN RESMI:

4. Analisa hasil percobaan yang Anda lakukan.
   Analisa setiap hasil tampilannya.
   Pada Percobaan 1 point 3 buatlah pohon dari struktur file dan direktori
   Bila terdapat pesan error, jelaskan penyebabnya.

2. Kerjakan latihan diatas dan analisa hasil tampilannya.
3. Berikan kesimpulan dari praktikum ini.

MODUL PRAKTIKUM

SISTEM OPERASI

PERTEMUAN 4

Proses & Manajemen Proses

Universitas Esa Unggul

Jakarta – 2008

Praktikum 4

Proses dan Manajemen Proses

POKOK BAHASAN:

- ✓ Proses pada Sistem Operasi Linux
- ✓ Manajemen Proses pada Sistem Operasi Linux

TUJUAN BELAJAR:

Setelah mempelajari materi dalam bab ini, mahasiswa diharapkan mampu:

- ✓ Memahami konsep proses pada sis tem operasi Linux.
- ✓ Menampilkan beberapa cara menampilkan hubungan proses parent dan child.
- ✓ Menampilkan status proses dengan beberapa format berbeda.
- ✓ Melakukan pengontrolan proses pada shell.
- ✓ Memahami penjadwalan prioritas.

DASAR TEORI:

1 KONSEP PROSES PADA SISTEM OPERASI LINUX

Proses adalah program yang sedang dieksekusi. Setiap kali menggunakan utilitas sistem atau program aplikasi dari shell, satu atau lebih proses ”child” akan dibuat oleh shell sesuai perintah yang diberikan. Setiap kali instruksi dibe rikan 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/inittab`).

Beberapa tipe proses :

10. **Foreground**

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

7. **Batch**

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

8. **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

2 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 :

No Sinyal	Nama	Deskripsi
1	SIGHUP	Hangup, sinyal dikirim bila proses terputus, misalnya melalui putusnya hubungan modem
2	SIGINT	Sinyal interrupt, melalui ^C
3	SIGQUIT	Sinyal Quit, melalui ^\
9	SIGKILL	Sinyal Kill, menghentikan proses
15	SIGTERM	Sinyal terminasi software

3 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.

4 MENGONTROL PROSES PADA SHELL

Shell menyediakan fasilitas job control yang memungkinkan mengontrol beberapa job atau proses yang sedang berjalan pada waktu yang sama. Misalnya bila melakukan pengeditan file teks dan ingin melakukan interrupt pengeditan untuk mengerjakan hal lainnya. Bila selesai, dapat kembali (*switch*) ke editor dan melakukan pengeditan file teks kembali.

Job bekerja pada **foreground** atau **background**. Pada *foreground* hanya diper untukkan untuk satu job pada satu waktu. Job pada *foreground* akan mengontrol shell - menerima input dari keyboard dan mengirim output ke layar. Job pada background tidak menerima input dari terminal, biasanya berjalan tanpa memerlukan interaksi.

Job pada *foreground* kemungkinan dihentikan sementara (suspend), dengan menekan [Ctrl-Z]. Job yang dihentikan sementara dapat dijalankan kembali pada *foreground* atau *background* sesuai keperluan dengan menekan ”**fg**” atau ”**bg**”. Sebagaicatatan, menghentikan job seme ntara sangat berbeda dengan melakuakan interrupt job (biasanya menggunakan [Ctrl-C]), dimana job yang diinterrup akan dimatikan secara permanen dan tidak dapat dijalankan lagi.

5 MENGONTROL PROSES LAIN

Perintah ps dapat digunakan untuk menunjukkan semua proses yang sedang berjalan pada mesin (bukan hanya proses pada shell saat ini) dengan format :

```
ps –fae atau
ps -aux
```

Beberapa versi UNIX mempunyai utilitas sistem yang disebut `top` yang menyediakan cara interaktif untuk memonitor aktifitas sistem. Statistik secara detail

dengan proses yang berjalan ditampilkan dan secara terus-menerus di-*refresh* . Proses ditampilkan secara terurut dari utilitas CPU. Kunci yang berguna pada `top` adalah

`s` – set update frequency

`u` – display proses dari satu user

`k` – kill proses (dengan PID)

`q` – quit

Utilitas untuk melakukan pengontrolan proses dapat ditemukan pada sistem UNIX adalah perintah `killall`. Perintah ini akan menghentikan proses sesuai PID atau job number proses.

TUGAS PENDAHULUAN :

Jawablah pertanyaan-pertanyaan di bawah ini :

13. Apa yang dimaksud dengan proses ?
14. Apa yang dimaksud perintah untuk menampilkan status proses : `ps, pstree.`

    Sebutkan opsi yang dapat diberikan pada perintah ps

    Apa yang dimaksud dengan sinyal ? Apa perintah untuk mengirim sinyal ?

    Apa yang dimaksud dengan proses foreground dan background pada job control ?

    Apa yang dimaksud perintah-perintah penjadwalan prioritas : `top, nice, renice.`

PERCOBAAN :

1. Login sebagai user.
2. Download program C++ untuk menampilkan bilangan prima yang bernama `primes.`
3. Lakukan percobaan-percobaan di bawah ini kemudian analisa hasil percobaan.
4. Selesaikan soal-soal latihan.

Percobaan 1 : Status Proses

6. Pindah ke *command line terminal* (tty2) dengan menekan **Ctrl+Alt+F2** dan login ke terminal sebagai user.
7. 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 (*Sleepin* g) dan R (*Running*), COMMAND merupakan instruksi yang digunakan.

```
$ ps
```

8. Untuk melihat faktor/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
```

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

```
$ ps –u <user>
```

8. Mencari proses lainnya gunakan opsi a (*all*) dan au (*all user*)

```
$ ps –a
$ ps –au
```

11. **Logout** dan tekan **Alt+F7** untuk kembali ke mode grafis

Percobaan 2 : Menampilkan Hubungan Proses Parent dan Child

1. Pindah ke *command line terminal* (tty2) dengan menekan **Ctrl+Alt+F2** dan login ke terminal sebagai user.

12. Ketik **ps –eH** dan tekan **Enter**. Opsi **e** memilih semua proses dan opsi **H** menghasilkan tampilan proses secara hierarki. Proses child muncul dibawah proses parent. Proses child ditandai dengan awalan beberapa spasi.

```
$ ps -eH
```

13. Ketik **ps –e f** dan tekan **Enter**. Tampilan serupa dengan langkah 2. Opsi **–f** akan menampilkan status proses dengan karakter grafis (\ dan _)

```
$ ps –e f
```

14. Ketik **pstree** dan tekan **Enter**. Akan ditampilkan semua proses pada sistem dalam bentuk hirarki parent/child. Proses parent di sebelah kiri proses child. Sebagai contoh proses `init` sebagai parent (*ancestor*) dari semua proses pada sistem. Beberapa child dari `init` mempunyai child. Proses `login` mempunya i proses `bash` sebagai child. Proses `bash` mempunyaiproses child `startx`. Proses `startx` mempunyai child `xinit` dan seterusnya.

```
$ pstree
```

15. Ketik **pstree | grep mingetty** dan tekan **Enter**. Akan menampilkan semua proses `mingetty` yang berjalan pada system yang berupa *consolevirtual*. Selain menampikan semua proses, proses dikelompokkan dalam satubaris dengan suatu angka sebagai jumlah proses yang berjalan.

```
$ pstree | grep mingetty
```

16. Untuk melihat semua PID untuk proses gunakan opsi **–p**.

```
$ pstree –p
```

17. Untuk menampilk an proses dan ancestor yang tercetak tebal gunakan opsi **–h**.

```
$ pstree –h
```

Percobaan 3 : Menampilkan Status Proses dengan Berbagai Format

9. Pindah ke *command line terminal* (tty2) dengan menekan **Ctrl+Alt+F2** dan login ke terminal sebagai user.
10. Ketik **ps –e | more** dan tekan **Enter**. Opsi **-e** menampilkan semua proses dalam bentuk 4 kolom : PID, TTY, TIME dan CMD.

    ```
    $ ps –e | more
    ```

    Jika halaman penuh terlihat prompt `--More--` di bagian bawah screen, tekan **q** untuk kembali ke prompt perintah.
4. Ketik **ps ax | more** dan tekan **Enter**. Opsi **a** akan menampilkan semua proses yang dihasilkan terminal (TTY). Opsi **x** menampilkan semua proses yang tidak dihasilkan terminal. Secara logika opsi ini sama dengan opsi **–e** . Terdapa 5 kolom : PID, TTY, STAT, TIME dan COMMAND.

    ```
    $ ps ax | more
    ```

    Jika halaman penuh terlihat prompt `--More--` di bagian bawah screen, tekan **q** untuk kembali ke prompt perintah.
5. Ketik **ps –e f | more** dan tekan **Enter**. Opsi **–e f** akan menampilkan semua proses dalam format daftar penuh.

    ```
    $ ps ef | more
    ```

    Jika halaman penuh terlihat prompt `--More--` di bagian bawah screen, tekan **q** untuk kembali ke prompt perintah.
6. Ketik **ps –eo pid, cmd | more** dan tekan **Enter**. Opsi **–eo** akan menampilkan semua proses dalam format sesuai definisi user yaitu terdiri dari kolom PID dan CMD.

    ```
    $ ps –eo pid,cmd | more
    ```

    Jika halaman penuh erlihat prompt `--More--` di bagian bawah screen, tekan **q** untuk kembali ke prompt perintah.

20. Ketik **ps –eo pid,ppid,%mem,cmd | more** dan tekan **Enter**. Akan menampilkan kolom PID, PPID dan %MEM. PPID adalah proses ID dari proses parent. %MEM menampilkan persentasi memory system yang digunakan proses. Jika proses hanya menggunakan sedikit memory system akan dita mpilkan 0.
    `$ ps –eo pid,ppid,%mem,cmd | more`
21. **Logout** dan tekan **Alt+F7** untuk kembali ke mode grafis

Percobaan 4 : Mengontrol proses pada shell

1. Pindah ke *command line terminal* (tty2) dengan menekan **Ctrl+Alt+F2** dan login ke terminal sebagai user.
2. Gunakan perintah `yes` yang mengirim output `y` yang tidak pernah berhenti `$ yes`

    Untuk menghentikannya gunakan **Ctrl-C.**
3. Belokkan standart output ke `/dev/null$ yes > /dev/null`

    Untuk menghentikannya gunakan **Ctrl-C.**
4. Salah satu cara agar perintah `yes` tetap dijalankan tetapi shell tetap digunakan untuk hal yang lain dengan meletakkan proses pada *background* dengan menambahkan karakter & pada akhir perintah. `$ yes > /dev/null &`

    Angka dalam ”[ ]” merupakan **job number** diikuti PID.
5. Untuk melihat status proses gunakan perintah `jobs.$ jobs`

7. Untuk menghentikan job, gunakan perintah `kill` diikuti *job number* atau PID proses. Untuk identifikasi job number, diikuti prefix dengan karakter ”%”.

   `$ kill %<nomor job>` contoh : `kill %1`

4. Lihat status job setelah diterminasi `$ jobs`

SISTEM OPERASI

PERTEMUAN 5

Shell pada Linux (Bash)

Universitas Esa Unggul

Jakarta – 2008

Praktikum 5

Bekerja Dengan Bash Shell

POKOK BAHASAN:

- ✓ History pada Bash Shell
- ✓ Membuat Bash Shell Script

TUJUAN BELAJAR:

Setelah mempelajari materi dalam bab ini, mahasiswa diharapkan mampu:

- ✓ Memahami shell pada sistem operasi Linux.
- ✓ Menggunakan feature history pada Bash Shell.
- Mengubah feature history pada Bash Shell.
- Mengubah prompt shell.
- Melakukan konfigurasi Bash Shell untuk menjalankan skrip secara otomatis.
- Membuat dan mengeksekusi shell script sederhana melalui editor vi.
- Memahami job control.
- Memahami stack.
- Menggunakan alias.

DASAR TEORI:

1 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 promp **#**.

Beberapa macam shell :

9. `/bin/sh`

Bourne shell, dirancang oleh Steve Bourne dari AT&T

15. `/bin/csh`
    Dikembangkan oleh UNIX Berkeley yang dikenal dengan C -Shell
16. `/bin/bash`
    Kompatibel dengan Bourne Shell dan juga mengadaptasi kemampuan Korn-Shell.

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

2 PROFILE

Pada saat login, program akan menjalankan beberapa program yaitu :

7. **/etc/profile**
   Berisi shell script yang berlaku untuk seluruh pengguna Linux.
8. **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 sebe lum logout

Isi dari /etc/profile:

```
7.    System wide environment and startup programs
8.Functions and aliases go in /etc/bashrc

PATH="$PATH:/usr/X11R6/bin"
PS1="[\u@\h \W]\\$ "
umask 022

USER='id -un'
LOGNAME=$USER
MAIL="/var/spool/mail/$USER"

HOSTNAME='/bin/hostname'
HISTSIZE=1000
HISTFILESIZE=1000
```

```
Export PATH PS1 HOSTNAME HISTSIZE HISTFILESIZE USER LOGNAME MAIL
```

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 direktory

3 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 kontrool 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

4 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.

5 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*.

6 EDITOR vi

Vi adalah full screen editor, artinya editor tersebut dapat memanfaatkan fasilitas satu layar penuh. Vi mempunyai 2 buah modus, yaitu :

4. Command line
   Editor vi mengintepretasikan input sebagai instruksi untuk dieksekusi oleh editor, contoh seperti mencari teks, mengganti teks secara otomatis dan lainnya.
5. Editing
   Editor vi mengintepretasikan input sebagai teks yang akan dimasukkan ke dalam buffer editor. Pada bagian bawah layar akan tampil teks “INSERTING”.

Pada awal vi dijalankan, maka program memasuki command mode. Dengan menekan tombol “i” maka akan memasuki editing. Untuk kembali ke command mode, tekan tombol Esc.

Kunci-kunci teks editor vi dapat dilihat pada tabel sebagai berikut :

Kunci	Keterangan	
H	Pindah kursor ke kiri satu karakter	
J	Pindah kursor ke kanan satu karakter	
K	Pinda h kursor ke atas	
L	Pindah kursor ke bawah	
O	Menyisipkan teks (satu baris setelah posisi kursor)	Untuk keluar dari 5 model kunci *insert* di samping ini dan mengaktifkan kunci-kunci lain, maka kita harus menekan tombol Esc terlebih dahulu.
I	Menyisip kan teks (di sebelah kiri posisi kursor)	
A	Menyisipkan teks (di sebelah kanan posisi kursor)	
I (shift i)	Menyisipkan teks (di posisi awal baris)	

A (shift a)	Menyisipkan teks (di posisi akhir baris)
X	Menghapus 1 huruf (di sebelah kanan posisi k ursor)
Dw	Manghapus 1 kata (di sebelah kanan posisi kursor)
Dd	Menghapus 1 baris (di sebelah kanan posisi kursor)
Yy	Mengkopi 1 baris
2yy	Mengkopi 2 baris
P	(*Paste*) Menampilkan baris kalimat yang sudah dikopi dengan kunci yy
Cw	Mengganti 1 kata yang telah ditulis di sebelah kanan posisi kursor dengan kata lain
Cc	Mengganti 1 baris kalimat yang telah ditulis di sebelah kanan posisi kursor dengan kalimat lain
ctrl-b	Mundur satu layar
ctrl- f	Maju satu layar
ctrl-d	Maju setengah layar
B	Menggerakkan kursor ke kiri satu kata
W	Manggerakkan kursor ke kanan satu kata
^	Pergi ke awal baris
$	Pergi ke akhir baris
U	Membatalkan perintah yang terakhir kali
U	Membatalkan seluruh perubahan teks pada baris tempat kursor berada
:!	Keluar untuk sementara dari editor vi dan menjalankan perintah yang lain

:wq	Write dan quite, simpan berkas dan keluar
:q!	Keluar vi tanpa menyimpan
:se all	Menampilkan semua pilihan set status
:se nu	Menampilkan nomor baris pada kiri layar
/string	Mencari string ke arah depan
?string	Mencari string ke arah belakang
N	Meneruskan pencarian untuk arah yang sama
N	Meneruskan pencarian untuk arah yang berbeda

TUGAS PENDAHULUAN :

Jawablah pertanyaan-pertanyaan di bawah ini :

8. Apa yang dimaksud dengan shell dan sebuatkan shell yang ada di system operasi Linux.

   Apa yang dimaksud dengan profile pada Bash Shell.

   Apa yang Anda ketahui mengenai file `.bashrc`.

   Apa yang dimaksud dengan history pada Bash Shell. Apa kegunaan perintah history, sebutkan cara-cara untuk mengetahui history perintah-perintah yang pernah digunakan oleh user!

   Cobalah menggunakan editor vi untuk mengetik dan pahami perintah-perintah yang ada seperti yang terdapat pada dasar teori (untuk dilakukan, tidak perlu dijawab sebagai tugas pendahuluan). Perintah-perintah yang penting : insert huruf(kalimat), delete (per huruf, per kata dan per baris), simpan file dan keluar dari editori vi.

PERCOBAAN :

2. Login sebagai user.
3. Bukalah Console Terminal dan lakukan percobaan-percobaan di bawah ini kemudian analisa hasil percobaan.
4. Selesaikan soal-soal latihan.

Percobaan 1 : Profile

13. 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 diekseksi sesaat sebelum logout, berfungsi sebagai *house clearing jobs*, artinya membersihkan semuanya, misalnya menghapustemporary file atau job lainnya. Melihat file .bash_logout dengan instruksi

```
$ cat .bash_logout
```

Percobaan 2 : Menggunakan Feature History Bash

Bash shell menyimpan ”history” perintah yang digunakan sebelumnya. Anda dapat mengaksis history dalam beberapa cara. Cara paling mudah adalah menggunakan **Panah Atas.** Maka perintah sebelumnya akan ditampilkan.

2. Berikutnya, berikan Bash shell beberapa perintah untuk diingat. Masukkan perintah berikut dan tekan **Enter** pada setiap baris.

```
$ cd
$ ls –l /etc
$ ls –l
$ whoami
$ who
```

5. Untuk memeriksa apakah perintah ini ditambahkan pada history, dapat menggunakan perintah `history` untuk melihat semua perintah yang pernah dimasukkan.

```
$ history
```

4. Anda dapat memilih perintah sebelumnya dengan menggunakan **Panah Atas ,** tetapi hal ini tidak efisien untuk perintah yang semakin bertambah banyak. Cara yang mudah menggunkaan nomor pada perintah history atau mencarinya. Untuk memilih dan mengeksekusi perintah dengan nomor, masukkan kunci ! diikuti nomor perintah.

   `$ !<Nomor Perintah>` Contoh : !780

5. Anda dapat mencari perintah dengan menyertakan perintah yang diinginkan. Misalnya **!?etc?!** akan menjalankan perintah `ls -l /etc` yang sebelumnya digunakan.

   `$ !?etc?`

6. Kemudian gunakan perintah history, maka akan terlihat perintah `ls -l/etc` yang kedua dan bukan**!?etc?**

   `$ history`

   Apabila string tidka ditemukan pada perintha history maka akan terdapat pesan error.

   `$ !?wombat99?`

   Jika diketikkan **!who** maka yang dijalankan adalah perintah `who`. Tetapi bila Anda ketikkan **!whoa** maka yang dijalankan adalah perintah `whoami`.

   ```
   $ !who
   $ !whoa
   ```

9. Anda bisa menggantikant string pada perintah history, terutama pada perintah yang panjang. Misalnya ketik **cat /bin/bash | strings | grep shell | less** dan tekan **Enter**. Maka akan menampilkan semua string pada file `/bin/bash` yang berisi kata ”shell”. Untuk keluar tekan q. Jika ingin menampilkan kata ”alias”, maka Anda tidak perlu mengetik perintah yang panjang lagi, tetapi cukup ketik ^**shell**^**alias**^ dan tekan **Enter** maka akan menggantikan kata ”shell” dengan ”alias”.

   ```
   $ cat /bin/bash | strings | grep shell |
   less $ ^shell^alias^
   ```

Percobaan 3 : Mengubah Feature History Bash

1. Bash shell akan menyimpan perintah history meskipun telah log out dan log in kembali. File `.bash_history` menyimpan file history yang terdapat pada home directory.
   ```
   $ cd
   ```
2. Lihat beberapa baris pada file .bash_history dengan ketik **tail .bash_history** dan tekan **Enter**. File ini bukan file yang up to date.
   ```
   $ tail .bash_history
   ```
3. Ketik **history** dan tekan **Enter**. Maka akan terlihat baris terakhir adalah perintah history dan baris sebelumnya adalah `tail .bash_history`. Perintah history bersifat up to date, karena disimpan pada memory sistem.
   ```
   $ history
   ```
4. Ketik perintah berikut
   ```
   $ echo 'Ini perintah saya'
   ```
5. Log out dan log in kembali sebagai user yang sama. Ketik **history** dan tekan **Enter**. Maka perintah`echo 'Ini perintah saya'`akan berada padabaris terakhir. Lihat file `.bash_history`, maka perintah tsb akan terdapat pada file `.bash_history`.
   ```
   $ history
   $ tail .bash_history
   ```
6. Ketik **history|less** untuk melihat perintah history terakhir pada screen. Tekan spacebar untuk melihat file lebih banyak. Untuk keluar tekan q
   ```
   $ history|less
   ```
7. Untuk melihat berapa banyak perintah history yang ada pada file ketik berikut dan output yang keluar serupa di bawah ini
   ```
   $ wc –l .bash_history
   1000       .bash_history
   ```

8. Output menunjukkan bahwa 1000 perintah history disimpan pada file history. Untuk melihat jangkauan (limit) perintah history digunakan variabel **HISTSIZE**. Untuk melihat jangkauan history ketik sebagai berikut
   ```
   $ set|grep HISTSIZE
   ```
9. Bila ingin memperbesar jangkauan file history, maka ubahlah variabel **HISTSIZE** pada skrip startup yang disebut .bashrc pada home directory.
   ```
   $ echo 'HISTSIZE=5000' >> .bashrc
   ```
10. Log out dan log in kembali sebagai user yang sama. Lihat perubahan variabel **HISTSIZE**.
    ```
    $ set|grep HISTSIZE
    ```
11. Ketikkan perintah history beberapa kali, maka perintah ini akan disimpan pada BASH history meskipun yang diketikkan perintahnya sama.
12. Anda dapat melakukan konfigurasi BASH agar tidak menambah perintah ke history jika perintah yang diketikkan sama dengan sebelumnya. Hal ini dilakukan dengan menambahkan variabel **HISTCONTROL** dan diberikan nilai **ignoredups** pada file `.bashrc`
    ```
    $ echo 'HISTCONTROL=ignoredups' >> .bashrc
    ```
13. Log out dan log in kembali sebagai user yang sama . Ketikkan history beberapa kali dan perhatikan berapa kali history muncul.

Percobaan 4 : Mengubah Prompt Shell

1. Prompt Bash shell dikonfigurasi dengan men-setting nilai variabel PS1. Selain menampilkan string statik sebagai prompt, Anda dapat menampilkan menjadi dinamis. Contohnya, apabila ingin menunjukkan *current directory* atau *current time*. Ketik **PS1='\t:'** dan tekan **Enter** untuk menampilkan waktu sistem da lam format 24 jam sebagai prompt Bash. Format dalam HH:MM:SS
   ```
   $ PS1='\t:'
   ```

3. Untuk menampilkan format 12 jam dengan indikator am dan pm ketik sebagai berikut :

```
$ PS1='\t:'
```

4. Kebanyakan orang menginginkan prompt Bash menampilkan *currentworking directory*. Direktory dapat ditampilkan dalam bentuk keseluruhanpath atau hanya nama direktory. Karakter `\w` menampilkan hanya nama direktory. Jika *current directory* adalah home directory, maka tampil prompt ~:

```
$ PS1='\w:'
```

5. Ketik **cd /usr/sbin** untuk melihat prompt `/usr/sbin:`

```
$ cd /usr/sbin
```

5. Ketik **PS1='\W:'** untuk melihat prompt

```
sbin:$ PS1='\W:'
```

6. Ada beberapa prompt BASH lain yang dapat diubah, yaitu PS2, PS3 dan PS4. Prompt PS2 digunakan sebagai prompt sekunder. Untuk melihat bagaimana penggunaannya, ketik **echo 'Hello** (tanpa diakhiri penutup quote) dan tekan **Enter**. Simbol lebih besar dari (>) akan muncul. Hal ini memberitahukanbahwa BASH menunggu Anda menyelesaikan perintah. Ketik penutup quote (**'**) dan tekan **Enter**. Perintah ini akan menyelesaikan prompt PS2, kata **"Hello, "** muncul diikuti dengan prompt PS1 pada baris baru.

```
$ echo
'Hello>'
```

7. Anda dapat mengubah prompt PS2 seperti mengubah prompt PS1. Ketik perintah berikut :

```
$ PS2='Selesai memasukkan perintah Anda:'
```

8. Kemudian ketik echo ’Hello (tanpa diakhiri penutup quote) dan tekan Enter. Pada baris berikutnya akan muncul **Selesai memasukkan perintah Anda:**. Kemudian ketikkan penutup quote (’) dan tekan **Enter**. Jika perintah selesai, maka kata **Hello** akan muncul diikuti prompt PS1 pada baris baru.

```
$ echo ’Hello
Selesai memasukkan perintah Anda:’
```

9. Prompt BASH dapat ditampilkan berwar na dengan melakukan setting *color-setting string* . Sebagai contoh, prompt BASH di-set dengan`\w\$`, akanmenampilkan *current working directory* yang diikuti $ (atau # jika anda login sebagai root). Untuk setting warna menjadi biru ketikkan berikut :

```
$ PS1=’\033[0;34m\w\$ \033[0;37m’
```

10. Untuk mendapatkan prompt warna merah ketikkan berikut :

```
$ PS1=’\033[0;31m\w\$ \033[0;37m’
```

30=hitam, 31=merah, 32=hijau, 34=biru, 35=ungu, 36=cyan, 37=putih.

11. Bila menginginkan beberapa warna, ketikkan perintah berikut : $

```
PS1=’\033[0;31m\w\033[0;32m\$ \033[0;37m’
```

12. Anda bisa menampilkan atribut visual seperti lebih terang, berkedip dan warna kebalikannya. Untuk menampilkan prompt yang lebih terang, atribut control diganti 1, seperti perintah berikut :

```
$ PS1=’\033[1;34m\w\033[1;32m\$ \033[0;37m’
```

13. Untuk menampilkan prompt dengan warna berkebalikan, atribut control diganti 7, seperti perintah berikut :

```
$ PS1=’\033[7;34m\w\033[7;32m\$ \033[0;37m’
```

14. Untuk menampilkan prompt berkedip, atribut control diganti 5, seperti perintah berikut :

```
$ PS1=’\033[5;34m\w\033[5;32m\$ \033[0;37m’
```

Percobaan 5 : Menambahkan otomatisasi ke Prompt Shell

1. Pastikan Anda berada di home directory `$ cd ~`
2. Buatlah skrip sederhana untuk mengurut daftar file. Anda dapat menggunakan teks editor, tetapi karena hanya satu baris, gunakan perintah echo untuk membuat file.
   ```
   $ echo 'sort ~/list > ~/r13; mv ~/r13 ~/list' > ~/sorter
   ```
3. Buatlah file skrip diatas menjadi file executable
   ```
   $ chmod +x sorter
   ```
4. Jalankan program sorter diatas setiap shell Bash menampilkan prompt PS1. Untuk melakukannya, buatlah variable **PROMPT_COMMAND** dimana nilainya adalah nama dari program `sorter`.
   ```
   $ PROMPT_COMMAND=~/sorter
   ```
5. Ketikkan **echo 'John Smith:13001'>>list** dan tekan **Enter**. Jika file `list` tidak ada, akan dibuat secara otomatis, tetapi jika sudah ada, string 'John Smith:13001' aka h ditambahkan.
   ```
   $ echo 'John Smith:13001'>>list
   ```
6. Ketik **cat list** dan tekan **Enter.** Maka Anda akan melihat isi file `list`. Pada saat ini, file mungkin mempunyai hanya satu baris sehingga tidak dapat dilihat apakah file sudah terurut. `$ cat list`
7. Masukkan bebe rapa perintah serupa dengan point 5 tetapi dengan nama dan nomor yang berbeda. Kemudian ketik **cat list** dan tekan **Enter.**
   ```
   $ echo 'Anita:13002'>>list
   $ echo 'Samantha:13003'>>list
   $ echo 'Patrik:13004'>>list
   $ echo 'Sponse Bob:13005'>>list
   $ echo 'Lisa:13006'>>list
   $ echo 'Squid:13007'>>list
   ```

8. Apabila Anda tidak menginginkan Shell Bash menampilkan file terurut sepanjang waktu, Anda tidak perlu menambahkan variable `PROMPT_COMMAND=~/sorter` pada file konfigurasi seperti .bashrc. BilaAnda ingin BASH berhenti menjalankan program sorter, maka ketikkan variable **PROMPT_COMMAND=** dan tekan **Enter** atau log out dan login kembali.

```
$ PROMPT_COMMAND=
```

SISTEM OPERASI

PERTEMUAN 6

Pemrograman Shell

Universitas Esa Unggul

Jakarta – 2008

Praktikum 6

Pemrograman Shell

POKOK BAHASAN:

- ✓ Pemrograman Shell

TUJUAN BELAJAR:

Setelah mempelajari materi dalam bab ini, mahasiswa diharapkan mampu:

- ✓ Mempelajari elemen dasar shell script
- ✓ Membuat program shell interaktif
- ✓ Menggunakan parameter dalam program
- ✓ Mempelajari test kondisi serta operator logic yang terkait dengan instruksi test
- ✓ Mengenal variable built-in dari shell
- ✓ Membuat aplikasi dengan shell menggunakan konstruksi if-then-else
- ✓ Menggunakan struktur case – esac.
- ✓ Loop dengan while, for, do while.
- ✓ Membuat fungsi dan mengetahui cara memanggil fungsi tersebut.

DASAR TEORI:

1 SHELL SCRIPT

Shell script dibuat dengan editor teks (ASCII editor) dan umumnya diberikan ekstensi ".sh". Script selalu diawali dengan komentar, yang dimulai dengan tanda #, disambung dengan ! dan nama shell yang digunakan.

```
#!/bin/sh                ❿
# Program shell          ②
#
var1=x                   ③
var2=8
```

✓ Awal dari program shell, komentar awal ini akan dibaca oleh system, kemudian system mengaktifkan program shell (/bin/sh) yang tertera di situ. Program shell dapat dipilih, misalnya /bin/csh, /bin/ksh dan lainnya

Adalah komentar, sebagai dokumentasi, baris ini akan diabaikan oleh program shell

③ Penggunaan variable (assignment), tidak boleh ada spasi di antara nama variable dan konstanta

2 VARIABEL

Variable shell adalah variable yang dapat mempunyai nilai berupa nilai String. Tata penulisan variable adalah sebagai berikut :

```
nama_var = nilai_var
```

Variable harus dimulai dengan alfabet, disusul dengan alfanumerik dan karakter lain. Variabel dapat ditulis dalam huruf kecil atau huruf besar atau campuran keduanya. Shell membedakan huruf besar dan huruf kecil (*case sensitive*), contoh :

```
VPT=poltek
i=5
```

Pemberian nilai variable tidak boleh dipisahkan dengan spasi, karena shell akan menganggap pemisahan tersebut sebagai parameter, contoh :

```
VPT    =poltek        ##error
VPT=    poltek        ##error
```

Untuk melihat nilai/isi dari sebuah variable, gunakan tanda $ di depan nama variable tersebut. Pada shell, instruksi echo dapat menampilkan isi variable tersebut, contoh :

```
VPT=poltek
echo $VPT

Gaji=450000
echo $Gaji echo
$VPT $Gaji
```

Bila menggunakan string yang terdiri dari lebih dari satu kata, maka string tersebut harus berada dalam tanda kutip atau apostrof, contoh :

```
VPT=poltek
VPT2=”poltek elektronika ITS”
```

3 MEMBACA KEYBOARD

Nilai variable dapat diisi melalui keyboard (stdin) dengan instruksi `read`.

4 PARAMETER

Sebuah program shell dapat mempunyai parameter sebanyak 9 buah dan direpresentasikan melalui variable khusus yaitu variable $!, $2, $3, $4, $5, $6, $7, $8 dan $9. Nama program she ll (nama script) direpresentasikan melalui variable $0.

Jumlah parameter dinyatakan sebagai $#. Bila tidak memberikan parameter, maka nilai $# adalah 0.

Shell variable $* menyatakan seluruh string yang menjadi parameter / argumen sebuah script ($@ mempunyai arti yang sama). $$ menyatakan nomor proses id (pid) dari script yang dijalankan. Pid ini akan terus berubah (umumnya) menaik, setiap kali proses berjalan.

5 STATUS EXIT

Setiap program setelah selesai dieksekusi akan memberikan informasi melalui variable spesial $?. Indikasi yang diberikan adalah :

- Bila program berakhir dengan sukses, $? = 0
- Bila program berakhir dengan error, $?≠0

Nilai dari status exit dapat dilihat melalui instruksi echo $?

6 KONSTRUKSI IF

```
if instruksi-awal then
     instruksi1
     instruksi2
     .....
fi
```

`if` akan mengeksekusi instruksi-awal, dan exit status dari instruksi tersebut akanmenjadi kondisi. Bila 0, maka instruksi selanjutnyua masuk ke dalam blok `then`. Bila tidak 0, maka alur program diteruskan setelah kunci kata `fi`.

7 KONSTRUKSI IF THEN ELSE

```
if instruksi1
then
        instruksi1.1
        instruksi1.2
        ..................
else
        instruksi2.1
        instruksi2.2
        ..................
fi
```

Bila status exit tidak sama dengan 0, maka kondisi menjadi FALSE dan instruksi setelah else akan dijalankan.

8 INSTRUKSI TEST

Instruksi test digunakan untuk memeriksa kondisi dari sebuah ekspresi. Ekspresi terdiri dari factor dan operator yang dipisahkan oleh spasi. Hasil test akan memberikan nilai berupa status exit, yaitu 0 bila ekspresi sesuai, bila tidak maka hasil adalah ≠ 0.

- Operator untuk test

Operator	0 atau TRUE, jika
`string1 = string2`	Identical
`string1 != string2`	Not identical
`-n string`	String is not null
`-z string`	String is null

- Test untuk files dan directory

Test dapat dilakukan untuk memeriksa apakah file ada (Exist), dapat dibaca, dapat ditulis, kosong dan lainnya.

Operator	0 atau TRUE, jika
`-f namafile`	File ada, file biasa
`-d namafile`	File ada, file adalah direktori
`-r namafile`	File dapat dibaca

`-w namafile`	File dapat ditulis
`-x namafile`	File adalah executable
`-s namafile`	File ada dan tidak kosong
`-w namafile`	File dapat ditulis

Untuk memudahkan pembacaan (readability), test dapat ditulis dengan

```
[ ekspresi ]
```

`[` sebenarnya adalah nama lain dari test, bedanya `[` akan mencari kurungpenutup `]` pada akhir ekspresi yang harus dipisahkan oleh spasi.

9 LOGICAL && DAN || (SHELL LEVEL)

Notasi && dan || digunakan untuk menggabungkan instruksi shell sebagai alternatif untuk if then else. Notasi && dan || sering ditemukan dalah shell script system administrator untuk menjalankan routine dari system operasi.

- `instruksi1 && instruksi2`
  shell akan mengeksekusi `instruksi1`, dan bila exit status `instruksi1` adalah FALSE, maka hasil dari AND tersebut sudah pasti sama dengan FALSE, sehingga `instruksi2` tidak mempunyai pengaruh lagi. Oleh karena itu, `instruksi2` tidak dijalankan. Sebaliknya bila hasil `instruksi1` adalah TRUE(0), maka `instruksi2` dijalankan
- `instruksi1 || instruksi2`
  shell akan menge ksekusi `instruksi1`, bila exit status adalah TRUE(0), hasil dari operasi OR tersebut sudah pasti menghasilkan TRUE, terlepas dari hasil eksekusi `instruksi2`. Oleh karena itu `instruksi2` tidak perlu dijalankan. Bila hasil `instruksi1` adalah FALSE, maka `instruksi2` akan dijalankan.

10 OPERATOR BILANGAN BULAT UNTUK TEST

Untuk membandingkan 2 buah bilangan, test memerlukan operator yang berbeda dengan string.

Operator	0 atau TRUE, jika
`i1 -eq i2`	Bilangan sama
`i1 -ge i2`	Lebih besar atau sama dengan
`i1 -gt i2`	Lebih besar
`i1 -le i2`	Lebih kecil atau sama dengan
`i1 -lt i2`	Lebih kecil
`i1 -ne i2`	Bilangan tidak sama

11 OPERATOR LOGICAL (TEST LEVEL)

Logical operator terdiri dari AND, OR dan NOT. Operator ini menggabungkan hasil ekspresi sebagai berikut :

NOT : symbol **!**

	!
True	False
False	True

AND : symbol **-a**

V1	V2	V1 –a V2
False	False	False
False	True	False
True	False	False
True	True	True

OR : symbol **-o**

V1	V2	V1 –o V2
False	False	False
False	True	True
True	False	True
True	True	True

12 KONSTRUKSI IF THEN ELSE IF

```
if instruksi1
then
      instruksi1.1
      instruksi1.2
      ……..
elif instruksi2
then

      instruksi2.1
      instruksi2.2
      ……..
else
      instruksi3.1
      instruksi3.2
      ……..
fi
```

Bila status exit tidak sama dengan 0, maka kondisi menjadi FALSE dan instruksi setelah else akan dijalankan.

13 HITUNGAN ARITMETIKA

Tipe dari variable SHELL hanya satu yaitu STRING. Tidak ada tipe lain seperti Numerik, Floating, Boolean ata u lainnya. Akibatnya variable ini tidak dapat membuat perhitungan aritmetika, misalnya :

```
A=5
B=$A +1    ## error
```

UNIX menyediakan utilitas yang bernama **expr** yaitu suatu utilitas yang melakukan aritmetika sederhana.

14 INSTRUKSI EXIT

Program dapat dihentikan (terminated/selesai) dengan instruksi exit. Sebagai nilai default program tersebut akan memberikan status exit 0.

15 KONSTRUKSI CASE

Case digunakan untuk menyederhanakan pemakaian if yang berantai, sehingga dengan case, kondisi dapat dikelompokkan secara lo gis 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 denan esac dan pada setiap kelompok instruksi diakhiri dengan ;;. Pada akhir pilihan yaitu *) yang berarti adalah “default”, bila kondisi tidak memenuhi pola sebelumnya

16 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
```

17 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 FALSSE, atau program keluar dari blok while melalui exit atau break.

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

18 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 ⇒ :

19 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 adi luar fungsi, sehingga tidak terjadi isi variable berubah.

TUGAS PENDAHULUAN :

Sebagai tugas pendahuluan, bacalah dasar teori diatas kemudian buatlah program Shell untuk Latihan 1 sampai dengan 5.

PERCOBAAN :

5. Login sebagai user.
6. Bukalah Console Terminal dan lakukan percobaan-percobaan di bawah ini kemudian analisa hasil percobaan.
7. Selesaikan soal-soal latihan.

Percobaan 1 : Membuat shell script

1. Buatlah file prog01.sh dengan editor vi

```
$ vi
prog01.sh#!/bi
n/sh
# Program shell
#
var1=x
var2=8
```

7. Untuk menjalankan shell, gunakan notasi TITIK di depan nama program `$ . prog01.sh`

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

```
$ chmod +x prog01.sh
$ ./prog01.sh
```

Percobaan 2 : Variabel

1. Contoh menggunakan variable pada shell interaktif

```
$ VPT=poltek
$ echo $VPT
```

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

```
$ VPT2=poltek elektronika(Terdapat pesan error)
$ VPT2=”poltek elektronika”
$ echo VPT2
$ echo $VPT2
```

12. Menggabungkan dua variable atau lebih

```
$ V1=poltek
$ V2=':'
$  V3=elektronika
$ V4=$V1$V2$V3
$ echo $V4
```

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

```
$ echo $V3
$ echo $V3ITS
$ echo ${V3}ITS
```

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

```
$ CMD=who
$ $CMD
$ CMD="ls -l"
$ $CMD
```

14. Modifikasi file `prog01.sh` berikut

```
$ vi
prog01.sh#!/bi
n/sh V1=poltek
V2=':'
V3=elektronika
echo        "Pemrograman
shell" echo $V1$V2$V3
V3=ITS
echo $V1$V2 di $V3
```

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

```
$ . prog01.sh
$ prog01.sh          (Terdapat pesan error)
$ ./prog01.sh        (Terdapat pesan error)
$ chmod +x prog01.sh
$ ./prog01.sh
```

Percobaan 3 : Membaca keyboard

1. Menggunakan instruksi `read`

```
$ read
namaamir
$ echo $nama
```

2. Membaca nama dan alamat dari keyboard

```
$ vi
prog02.sh#!/bi
n/sh
14.     prog02.sh
15.     membaca nama dan alamat

echo "Nama Anda :
" read nama
echo "Alamat :
" read alamat
echo "Kota : "
read kota

echo
echo "Hasil adalah : $nama, $alamat di $kota"
```

17. Eksekusi program `prog02.sh`

```
$ . prog02.sh
Nama Anda :
Amir
Alamat :
Jl semangka 67
Kota :
Surabaya

Hasil adalah : Amir, Jl semangka di Surabaya
```

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

```
$ vi prog02.sh
#!/bin/sh
  prog02.sh
  membaca nama dan alamat
```

```
echo –n “Nama Anda :
“ read nama
echo –n “Alamat :
“ read alamat
echo –n “Kota : “
read kota
echo
echo “Hasil adalah : $nama, $alamat di $kota”
```

5. Eksekusi program `prog02.sh`

```
$ . prog02.sh
Nama Anda : Amir
Alamat : Jl semangka
67 Kota : Surabaya

Hasil adalah : Amir, Jl semangka di Surabaya
```

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

```
$ read
nama<CR>
$ echo
$nama $ A=
$ B=””
$ C=$A$B
$ echo $C
```

4. Variabel dapat disubtitusikan dengan hasil eksekusi dari sebuah instruksi. Pada contoh dibawah , instruksi pwd dieksekusi lebih dahulu dengan sepasang Back Quate (tanda kutip terbalik). Hasil dari eksekusi tersebut akan masuk sebagai nilai variable DIR

```
$ pwd
$ DIR=`pwd`
$ echo $DIR
```

8. Buatlah shell script `prog03.sh`

```
$ vi
prog03.sh#!/bi
n/sh
# prog03.sh
#
NAMA=`whoami`

echo Nama Pengguna Aktif adalah $NAMA

tanggal=`date | cut –c1-10`

echo Hari ini tanggal $tanggal
```

9. Eksekusi

```
prog03.sh$      .
prog03.sh
```

Percobaan 4 : Parameter

1. Membuat shell script `prog04.sh`

```
$ vi
prog04.sh#!/bi
n/sh
# prog04.sh versi 1
# Parameter passing
#
echo “Nama program adalah $0”
echo “Parameter 1 adalah $1”
echo “Parameter 2 adalah $2”
echo “Parameter 3 adalah $3”
```

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

```
$ . prog04.sh
$ . prog04.sh amir hasan
$ . prog04.sh amir hasan badu ali
```

3. Membuat shell script `prog04.sh` versi 2 dengan memberikan jumlah parameter

```
$ vi prog04.sh
#!/bin/sh
# prog04.sh versi 2
# Parameter passing
#
echo "Jumlah parameter yang diberikan adalah
$#" echo "Nama program adalah $0"
echo "Parameter 1 adalah $1"
echo "Parameter 2 adalah $2"
echo "Parameter 3 adalah $3"
```

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

```
$ . prog04.sh
$ . prog04.sh amir hasan badu ali
```

5. Membuat shell script `prog04.sh` versi 3 dengan menambahkan total parameter dan nomor proses id (PID)

```
$ vi prog04.sh
#!/bin/sh
# prog04.sh versi 3
# Parameter passing
#
echo "Jumlah parameter yang diberikan adalah
$#" echo "Nama program adalah $0"
echo "Parameter 1 adalah $1" echo
"Parameter 2 adalah $2" echo
"Parameter 3 adalah $3" echo
"Total parameter adalah $*"
echo "PID proses shell ini adalah $$"
```

6. Eksekusi `prog04.sh` dengan 4 parameter

```
$ . prog04.sh amir hasan badu ali
```

Percobaan 5 : Status Exit

1. String tidak diketemukan, maka status exit adalah 1

```
$ grep xyz
/etc/passwd $ echo $?
```

2. String diketemukan, maka status exit adalah 0

```
$ grep <user>
/etc/passwd $ echo $?
```

Percobaan 6 : Konstruksi if

1. Instruksi dengan exit status 0

```
$ who
$ who | grep
<user> $ echo $?
```

4. If membandingkan exit status dengan 0, bila sama, maka blok program masuk ke dalam blok then-fi

```
$ if [ $? = 0 ]
> then
>       echo “Pemakai tersebut sedang aktif”
> fi
```

3. Nomor (1) dan (2) diatas dapat disederhanakan dengan

```
$ if who|grep <user>>/dev/null
8. then
9.       echo okay
10.fi
```

Percobaan 7 : Konstruksi if then else

1. Membuat shell script `prog05.sh`

```
$ vi
prog05.sh#!/bi
n/sh
10.     prog05.sh
11.     Program akan memberikankonfirmasi apakah nama
12.     user sedang aktif atau tidak
#
echo –n “Berikan nama pemakai :
” read nama
if who | grep $nama >
/dev/null then
        echo “$nama sedang aktif”
else
        echo “$nama tidak aktif”
fi
```

2. Jalankan `prog05.sh`, masukkan nama pemakai yang aktif yang tampil pada instruksi who dan coba juga untuk nama pemakai yang tidak aktif

```
$ who
$ . prog05.sh[nama=<user>]
$ . prog05.sh[nama=studentOS]
```

MODUL PRAKTIKUM

SISTEM OPERASI

PERTEMUAN 7

System Call & Manajemen Memory

Universitas Esa Unggul

Jakarta – 2008

Praktikum 7

System Call dan Manajemen Memory

POKOK BAHASAN:

- ✓ UNIX System Call
- ✓ Manajemen Memory

TUJUAN BELAJAR:

Setelah mempelajari materi dalam bab ini, mahasiswa diharapkan mampu:

- ✓ Menggunakan system call fork, wait dan execl pada Linux.
- ✓ Menggunakan perintah-perintah untuk manajemen memory.

DASAR TEORI:

1 UNIX SYSTEM CALL

Pada praktikum ini akan dilakukan percobaan menggunakan system call yang berhubungan dengan proses pada system operasi UNIX yang biasa disebut UNIX System Call, yaitu system call fork, execl dan wait. Pada percobaan yang dilakukan akan dibuat program yang didalamnya terdapat fungsi system call. Untuk menjalankannya pada Linux gunakan g++.

System Call Fork

System call fork adalah suatu system call yang membuat suatu proses baru pada system operasi UNIX. Pada percobaan ini menggunakan mesin Linux dan beberapa program yang berisi system call `fork()`.

Bila suatu program berisi sebuah fungsi `fork()`, eksekusi dari program menghasilkan eksekusi dua proses. Satu proses dibuat untuk memulai eksekusi program. Bila system call `fork()` dieksekusi, proses lain dibuat. Proses asal disebut proses parend dan proses kedua disebut proses child. Proses child merupakan duplikat dari proses parent. Kedua proses melanjutkan eksekusi dari titik dimana system call `fork()` menghasilkan eksekusi pada program utama. Karena UNIX adalah systemoperasi time sharing, dua proses tersebut dapat mengeksekusi secara konkuren.

Nilai yang dihasilkan oleh `fork()` disimpan dalam variable bertipe `pid_t`, yang berupa nilai integer. Karena nilai dari variable ini tidak digunakan, maka hasil `fork()` dapat diabaikan.

18. Untuk kill proses gunakan **Ctrl+C.**
19. Untuk dokumentasi `fork()` dapat dilihat dengan ketikkan `man 2 fork`.
10. Untuk melihat id dari proses, gunakan system call `getpid()`
11. Untuk melihat dokumentasi dari `getpid()`, ketikkan `man 2 getpid`

Perbedaan antara proses parent dan proses child adalah

17. Mempunyai pid yang berbeda
18. Pada proses parent , `fork()` menghasilkan pid dari proses child jika sebuah proses child dibuat.
19. Pada proses child, `fork()` selalu menghasilkan 0
20. Membedakan copy dari semua data, termasuk variable dengan current value dan stack
21. Membedakan program counter (PC) yang menunjukkan eksekusi berikutnya meskipun awalnya keduanya mempunyai nilai yang sama teta pi setelah itu berbeda.
22. Setelah fork, kedua proses tersebut tidak menggunakan variable bersama.

System call fork menghasilkan :

9. Pid proses child yang baru ke proses parent, hal ini sama dengan memberitahukan proses parent nama dari child-nya
10. 0 : menunjukkan proses child
11. -1 : 1 jika terjadi error, `fork()` gagal karena proses baru tidak dapat dibuat.

System Call Wait

System call wait menyebabkan proses menunggu sinyal (menunggu sampai sembarang tipe sinyal diterima dari sembarang proses). Biasanya digunakan oleh proses parent untuk menunggu sinyal dari system operasi ke parent bila child diterminasi.

System call wait menghasilkan pid dari proses yang mengirimi sinyal. Untuk melihat dokumentasi wait gunakan perintah `man 2 wait`.

System Call Execl

Misalnya kita ingin proses baru mengerjakan sesuatu yang berbeda dari proses parent, sebutlah menjalankan program yang berbeda. Sistem call execl meletakkan program executable baru ke memory dan mengasosiasikannya dengan proses saat itu. Dengan kata lain, mengubah segala sesuatunya sehingga program mulai mengeksekusi dari file yang berbeda.

2 MANAJEMEN MEMORY

Linux mengimplementasikan sistem virtual memory demand-paged. Proses mempunyai besar memory virtual yang besar (4 gigabyte). Pada virtual memory dilakukan transfer page antara disk dan memory fisik.

Jika tidak terdapat cukup memory fisik, kernel melakukan swapping beberapa page lama ke disk. Disk drive adalah perangkat mekanik yang membaca dan menulis ke disk yang lebih lambat dibandingkan mengakses memory fisik. Jika memory total page lebih dari memory fisik yang tersedia, kernel lebih banyak melakukan swapping dibandingkan eksekusi kode program, sehingga terjadi thrashing dan mengurangi utilitas.

Jika memory fisik ekstra tidak digunakan, kernel meletakkan kode program sebagai disk buffer cache. Disk buffer menyimpan data disk yang diakses di memory; jika data yang sama dibutuhkan lagi dapat dengan cepat diambil dari cache.

Pertama kali sistem melakukan booting, ROM BIOS membentuk memory test seperti terlih at berikut :

```
ROM BIOS (C) 1990
008192 KB OK WAIT......
```

Kemudian informasi penting ditampilkan selama proses booting pada linux seperti terlihat berikut :

```
Memory: 7100k/8192k available (464k kernel
code, 384k reserved, 244k data) ...
Adding Swap: 19464k swap-space
```

Informasi diatas menampilkan jumlah RAM tersedia setelah kernel di-load ke memory (dalam hal ini 7100K dari 8192K). Jika ingin melihat pesan saat booting kernel yang terlalu cepat dibaca dapat dilihat kembali dengan perintah `dmesg`.

Setiap Linux dijalankan, perintah `free` digunakan untuk menampilkan total memory yang tersedia. Atau menggunakan `cat /proc/meminfo`. Memory fisik dan ruang swap ditampilkan disini. Contoh output pada sistem :

```
total used free shared buffers
Mem:   7096 5216 1880   2328    2800
Swap: 19464 0    19464
```

Informasi ditampilkan dalam kilobyte (1024 byte). Memory ”`total`” adalah jumlah tersedia setelah load kernel. Memory digunakan untuk proses atau disk bufferring sebagai “`used`”. Memory yang sedang tidak digunakan ditampilkan pada kolom “`free`”. Memory total sama dengan jumlah kolom ”`used`” dan ”`free`”.

Memory diindikasikan “`shared`” yaitu berapa banyak memory yang digunakan lebih dari satu proses. Program seperti shell mempunyai lebih dari satu proses yang berjalan. Kode executable read-only dan dapat disharing oleh semua proses yang berjalan pada shell. Kolom “`buffers`” menampilkan berapa banyak memory digunakan untuk disk buffering.

Perintah `free` juga menunjukkan dengan jelas bagaimana swap space dilakukan dan berpa banyak swapping yang terjadi.

Percobaan berikut untuk mengetahui manajemen memory :

1. Pada saat bootup, dengan satu user log in, dengan perintah `free` sistem melaporkan berikut :

```
             total       used       free     shared    buffers     cached
Mem:        247184     145772     101412          0      10872      57564
-/+ buffers/cache:      77336     169848
Swap:       522072          0     522072
```

   Terdapat free memory (4.4MB) dan sedikit disk buffer (1.1MB).

2. Situasi berubah setelah menjalankan perintah yang membaca data dari disk (command `ls –lR /`.)

```
             total       used       free     shared    buffers     cached
Mem:        247184     230604      16580          0      45260      59748
-/+ buffers/cache:     125596     121588
Swap:       522072        308     522072
```

   Disk buffer bertambah menjadi 2 MB. Hal ini berakibat pula pada kolom ”`used`” dan memory ”`free`” juga berkurang.

Perintah `top` dan `ps -u` juga sangat berguna untuk menunjukkan bagaimana penggunaan memory berubah secara dinamis dan bagaimana proses individu menggunakan memory. Contoh tampilannya :

```
USER     PID %CPU %MEM    VSZ RSS TTY      STAT START   TIME COMMAND
student 4581  0.0  0.3   4316 856 pts/0    S     10:25   0:00 bash
student 4699  0.0  0.2   2604 656 pts/0    R     10.39   0:00 ps -u
```

TUGAS PENDAHULUAN :

Jawablah pertanyaan-pertanyaan berikut ini :

9. Apa yang dimaksud dengan system call ?
10. Apa yang dimaksud dengan sistem call `fork(), execl()` dan `wait().` Jawablah dengan menggunakan perintah man (contoh : `man 2 fork`, `man 2 execl` dan `man 2 wait`) ?
6. Apa yang dimaksud sistem virtual memory, proses swapping dan buffer cache pada manajemen memory ?
7. Apa yang dimaksud perintah free dan cat /proc/meminfo ?
8. Apa yang dimaksud perintah ps ?

PERCOBAAN:

9. Login sebagai user.
10. Bukalah Console Terminal dan lakukan percobaan-percobaan di bawah ini kemudian analisa hasil percobaan.
11. Selesaikan soal-soal latiha n.

Percobaan 1 : Melihat proses parent dan proses child

12. Dengan menggunakan editor vi, buatlah file `fork1.cpp` dan ketikkan program berikut :

```
#include <iostream>
using namespace std;
#include <sys/types.h>
#include <unistd.h>

/* getpid() adalah system call yg dideklarasikan pada
   unistd.h. Menghasilkan suatu nilai dengan type pid_t.
   pid_t adalah type khusus untuk process id yg ekuivalen dg int
*/

int main(void) {
   pid_t  mypid;
   uid_t myuid;

   for (int i = 0; i < 3; i++) {
      mypid = getpid();
      cout << "I am process " << mypid << endl;
      cout << "My parent is process " << getppid() << endl;
      cout << "The owner of this process has uid " << getuid()
           << endl;
      /* sleep adalah system call atau fungsi library
         yang menghentikan proses ini dalam detik
      */
            sleep(1);
   }
   return 0;
}
```

2. Gunakan g++ compiler untuk menjalankan program diatas

```
$ g++ -o fork1
fork1.cpp $ ./fork1
```

3. Amati output yang dihasilkan

Percobaan 2 : Membuat dua proses terus menerus dengan sebuah system call fork()

14. Dengan menggunakan editor vi, buatlah file `fork2.cpp` dan ketikkan program berikut :

```
#include <iostream>
using namespace std;
#include <sys/types.h>
#include <unistd.h>

/* getpid() dan fork() adalah system call yg
   dideklarasikan pada unistd.h.
   Menghasilkan suatu nilai dengan type pid_t.
   pid_t adalah type khusus untuk process id yg ekuivalen dg int
*/

int main(void) {
   pid_t childpid;
   int x = 5;
   childpid = fork();
   while (1) {
       cout << "This is process " << getpid() <<
       endl; cout << "x is " << x << endl;
       sleep(1);
       x++;
   }
   return 0;
}
```

5. Gunakan g++ compiler untuk menjala nkan program diatas. Pada saat dijalankan, program tidak akan pernah berhenti. Untuk menghentikan program tekan **Ctrl+C.**

```
$ g++ -o fork2
fork2.cpp $ ./fork2
```

3. Amati output yang dihasilkan

Percobaan 3 : Membuat dua proses sebanyak lima kali

8. Dengan menggunakan editor vi, buatlah file `fork3.cpp` dan ketikkan program berikut :

```
#include <iostream>
using namespace std;
#include <sys/types.h>
#include <unistd.h>
```

```
/* getpid() dan fork() adalah system call yg
   dideklarasikan pada unistd.h.
   Menghasilkan suatu nilai dengan type pid_t.
   pid_t adalah type khusus untuk process id yg ekuivalen dg int
*/
int main(void) {
   pid_t childpid;
   childpid = fork();
   for (int i = 0;  i < 5;  i++) {
       cout << "This is process " << getpid() <<
       endl; sleep(2);
   }
   return 0;
}
```

2. Gunakan g++ compiler untuk menjalankan program diatas

```
$ g++ -o fork3
fork3.cpp $ ./fork3
```

3. Amati output yang dihasilkan

Percobaan 4 : Proses parent menunggu sinyal dari proses child dengan system call wait

21. Dengan menggunakan editor vi, buatlah file `fork4.cpp` dan ketikkan program berikut :

```
#include <iostream>
using namespace std;
#include <sys/types.h>
#include <unistd.h>
#include <sys/wait.h>

/* pid_t fork() dideklarasikan pada unistd.h.
   pid_t adalah type khusus untuk process id yg ekuivalen dg int
*/
```

```
int main(void) {
    pid_t child_pid;
    int status;
    pid_t wait_result;

    child_pid = fork();
    if (child_pid == 0) {
        /* kode ini hanya dieksekusi proses child */
        cout << "I am a child and my pid = " << getpid() <<
        endl; cout << "My parent is " << getppid() << endl;
        /* keluar if akan menghentikan hanya proses child */
    }
    else if (child_pid > 0) {
        /* kode ini hanya mengeksekusi proses parent */ cout
        << "I am the parent and my pid = " << getpid()
             << endl;
        cout << "My child has pid = " << child_pid << endl;
    }
    else {
        cout << "The fork system call failed to create a
                 new process" << endl;
        exit(1);
    }

    /* kode ini dieksekusi baik oleh proses parent dan child
    */ cout << "I am a happy, healthy process and my pid = "
         << getpid() << endl;

    if (child_pid == 0) {
        /* kode ini hanya dieksekusi oleh proses child */
        cout << "I am a child and I am quitting work now!"
             << endl;
    }
    else {
        /* kode ini hanya dieksekusi oleh proses parent */
        cout << "I am a parent and I am going to wait for my
                  child" << endl;
        do {
            /* parent menunggu sinyal SIGCHLD mengirim
               tanda bahwa proses child diterminasi */
           wait_result = wait(&status);
        } while (wait_result != child_pid);
        cout << "I am a parent and I am quitting." << endl;
    }
    return 0;
}
```

2. Gunakan g++ compiler untuk menjalankan program diatas

```
$ g++ -o fork4
fork4.cpp $ ./fork4
```

3. Amati output yang dihasilkan

Percobaan 5 : System call fork/exec dan wait mengeksekusi program bernama ls, menggunakan file executable /bin/ls dengan satu parameter –l yang ekuivalen dengan ls –l

1. Dengan menggunakan editor vi, buatlah file `fork5.cpp` dan ketikkan program berikut :

```
#include <iostream>
using namespace std;
#include <sys/types.h>
#include <unistd.h>
#include <sys/wait.h>

/* pid_t fork() dideklarasikan pada unistd.h.
   pid_t adalah type khusus untuk process id yg ekuivalen dg int
*/

int main(void) {
   pid_t child_pid;
   int status;
   pid_t wait_result;

   child_pid = fork();
   if (child_pid == 0) {
       /* kode ini hanya dieksekusi proses child */
       cout << "I am a child and my pid = " << getpid() <<
       endl; execl("/bin/ls", "ls", "-l", "/home", NULL);
       /* jika execl berhasil kode ini tidak pernah digunakan
       */ cout << "Could not execl file /bin/ls" << endl;
       exit(1);
       /* exit menghentikan hanya proses child */
   }
   else if (child_pid > 0) {
       /* kode ini hanya mengeksekusi proses parent */ cout
       << "I am the parent and my pid = " << getpid()
            << endl;
       cout << "My child has pid = " << child_pid << endl;
   }
   else {
       cout << "The fork system call failed to create a
                 new process" << endl;
       exit(1);
   }
```

```
    /* kode ini hanya dieksekusi oleh proses parent karena
    child mengeksekusi dari "/bin/ls" atau keluar   */ cout
       << "I am a happy, healthy process and my pid = "
        << getpid() << endl;

    if (child_pid == 0) {
       /* kode ini tidak pernah dieksekusi */
       printf("This code will never be executed!\n");
    }
    else {
       /* kode ini hanya dieksekusi oleh proses parent */
       cout << "I am a parent and I am going to wait for my
             child" << endl;
       do {
          /* parent menunggu sinyal SIGCHLD mengirim
             tanda bila proses child diterminasi */
          wait_result = wait(&status);
       } while (wait_result != child_pid);
       cout << "I am a parent and I am quitting." << endl;
    }
    return 0;
}
```

2. Gunakan g++ compiler untuk menjalankan program diatas

```
$ g++ -o fork5
fork5.cpp $ ./fork5
```

3. Amati output yang dihasilkan

Percobaan 6 : System call fork/exec dan wait mengeksekusi program lain

1. Dengan menggunakan editor vi, buatlah file `fork6.cpp` dan ketikkan program berikut :

```
#include <iostream>
using namespace std;
#include <sys/types.h>
#include <unistd.h>
#include <sys/wait.h>

/* pid_t fork() dideklarasikan pada unistd.h.
   pid_t adalah type khusus untuk process id yg ekuivalen dg int
*/
```

```
int main(void) {
    pid_t child_pid;
    int status;
    pid_t wait_result;

    child_pid = fork();
    if (child_pid == 0) {
        /* kode ini hanya dieksekusi proses child */
        cout << "I am a child and my pid = " << getpid() <<
        endl; execl("fork3", "goose", NULL);
        /* jika execl berhasil kode ini tidak pernah digunakan
        */ cout << "Could not execl file fork3" << endl;
        exit(1);
        /* exit menghentikan hanya proses child */
    }
    else if (child_pid > 0) {
        /* kode ini hanya mengeksekusi proses parent */ cout
        << "I am the parent and my pid = " << getpid()
             << endl;
        cout << "My child has pid = " << child_pid << endl;
    }
    else {
        cout << "The fork system call failed to create a
                  new process" << endl;
        exit(1);
    }

     /* kode ini hanya dieksekusi oleh proses parent karena
      child mengeksekusi dari "fork3" atau keluar   */ cout
        << "I am a happy, healthy process and my pid = "
         << getpid() << endl;

    if (child_pid == 0) {
       /* kode ini tidak pernah dieksekusi */
       printf("This code will never be executed!\n");
    }
    else {
       /* kode ini hanya dieksekusi oleh proses parent */
       cout << "I am a parent and I am going to wait for my
               child" << endl;
       do {
          /* parent menunggu sinyal SIGCHLD mengirim
             tanda bila proses child diterminasi */
          wait_result = wait(&status);
       } while (wait_result != child_pid);
       cout << "I am a parent and I am quitting." << endl;
    }
    return 0;
}
```

2. Gunakan g++ compiler untuk menjalankan program diatas

```
$ g++ -o fork6
fork6.cpp $ ./fork6
```

3. Amati output yang dihasilkan

Percobaan 7 : Melihat Manajemen Memory

6. Perhatikan dengan perintah `dmesg` jumlah memory tersedia dan proses swapping

```
$ dmesg | more
```

7. Dengan perintah `free` perhatikan jumlah memory ”free”, ”used”, “share” dan “buffer” .

```
$ free
```

8. Dengan perintah dibawah ini apakah hasilnya sama dengan no 2

```
? $ cat /proc/meminfo
```

9. Gunakan perintah dibawah ini

```
$ ls –lR /.
```

10. Perhatikan perubahan manajemen memory

```
$ free
```

11. Jalankan sebuah program, misalnya open Office. Perhatikan perubahan manajemen memory

```
$ free
```

7. Dengan perintah ps bagaimana penggunaan memory untuk se tiap proses diatas ?

```
$ ps -uax
```

LATIHAN:

5. Ubahlah program fork5.cpp pada percobaan 5 untuk mengeksekusi perintah yang ekuivalen dengan
   a. `ls –al /etc.`
   b. `cat fork2`
   c. `./fork2`
7. Informasi apa saja mengenai manajemen memory yang ditampilkan pada perintah `dmesg` pada percobaan Anda ?
8. Bagaimana informasi yang ditampilkan dengan perintah `free` pada percobaan Anda ?
9. Apa isi file `/proc/meminfo` pada percobaan yang Anda lakukan ?
10. Berapa besar memory yang digunakan setelah percobaan 7 dengan perintah `ps –uax` ?
11. Lakukan hal yang sama dengan percobaan 7 untuk melihat perubahan memory setelah dilakukan beberapa proses pada shell. Tentukan perintah yang dilakukan misalnya membuka browser dan perhatikan hal-hal berikut :
    Informasi apa saja yang ditampilkan dengan perintah `free` ?
    Informasi apa saja yang disimpan file `/proc/meminfo` ?
    Berapa besar kapasitas memory total ?
    Berapa kapasitas memory yang sudah terpakai ?
    Berapa kapasitas memory yang belum terpakai ?
    Berapa kapasitas memory yang digunakan sharing beberapa proses ?
    Berapa kapasitas buffer cache ?

LAPORAN RESMI:

10. Analisa hasil percobaan yang Anda lakukan.
11. Kerjakan latihan diatas dan analisa hasil tampilannya.
12. Berikan kesimpulan dari praktikum ini.

SISTEM OPERASI

PERTEMUAN 8

Sistem File

Universitas Esa Unggul

Jakarta – 2008

Praktikum 8

Sistem File

POKOK BAHASAN:

- ✓ Sistem file

TUJUAN BELAJAR:

Setelah mempelajari materi dalam bab ini, mahasiswa diharapkan mampu:

- ✓ Memahami atribut file dan ijin akses.
- ✓ Memahami perintah untuk mengubah ijin akses suatu file.
- ✓ Menggunakan perintah-perintah untuk mengubah ijin akses..

DASAR TEORI:

1 ATRIBUT FILE

File mempunyai beberapa atribut, antara lain :

- Tipe file : menentukan tipe dari file, yaitu :

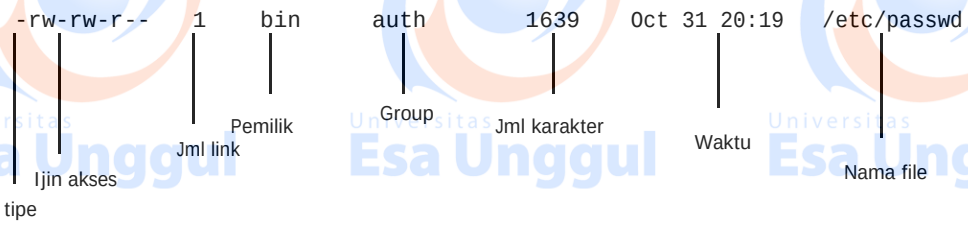
Karakter	Arti
-	File biasa
d	Direktori
l	Symbolic link
b	Block special file
c	Character special file
s	Socket link
p	FIFO

- 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 :



2 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 :

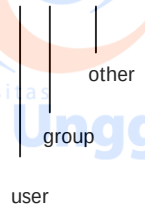
20. Pemilik (Owner)
21. Kelompok (Group)
22. Lainnya (Others)

Setiap peran dapat melakukan 3 bentuk operasi yaitu :

12. Pada File
    - R (Read) Ijin untukmembaca
    - W (Write) Ijin untuk mengubah / membuat
    - X (Execute) Ijin untuk menjalankan program
23. Pada Direktori
    - R (Read) Ijin untukmembaca 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
```



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 = all

Format lain dari chmod adalah menggunakan bilangan octal sebagai berikut

```
r    w    x
4    2    1      =    7
```

3 USER MASK

Untuk menentukan ijin 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

TUGAS PENDAHULUAN :

Sebagai tugas pendahuluan, jawablah pertanyaan-pertanyaan berikut ini :

12. Apa saja atribut file ? Berikan contoh file dengan tipe file yang disebutkan pada dasar teori.
13. Apa yang dimaksud ijin akses ? Bagaimana contoh penggunaan perintah chmod untuk mengubah ijin akses.
14. Berilah contoh penggunaan perintah umask untuk mengubah ijin akses.

PERCOBAAN :

11. Login sebagai user.
12. Bukalah Console Terminal dan lakukan percobaan-percobaan di bawah ini kemudian analisa hasil percobaan.
13. Selesaikan soal-soal latihan.

Percobaan 1 : Ijin Akses

9. Melihat identitas diri melalui `etc/passwd` atau `etc/group`, informasi apa ditampilkan ?

```
$ id
$ grep <user> /etc/passwd
$ grep[Nomor group id]/etc/group
```

2. Memeriksa direktori home

```
$ ls -ld /home/<user>
```

3. Mengubah Ijin akses (chmod). Perhatikan ijin akses setiap perubahan !

```
$ touch f1 f2
f3 $ ls -l
$  chmod  u+x
f1 $ ls –l f1
$  chmod  g=w
f1 $ ls –l f1
$  chmod  o-r
f1 $ ls –l f1
$  chmod  a=x
f2 $ ls –l f2
$ chmod u+x,g-r,o=w
f3 $ ls –l f3
$ chmod 751 f1
$ chmod 624 f2
$ chmod 430 f3
$ ls –l f1 f2 f3
```

4. Mengganti kepemilikan digunakan perintah chown. Masuk ke root untuk mengganti kepemilikan tersebut.

```
$ su root
$ echo Hallo >
f1 $ ls –l f1
$ chown <user-baru> f1contoh :chown student1 f1
$ ls –l f1
```

12. Ubahlah ijin akses home directory <user> (student) pada root sehingga <user-baru>(student1) pada satu group dapat mengakses home direktory<user>.

Hal ini dimaksudkan agar file f1 yang sudah diubah kepemilikannya dapat diakses <user-baru>. Perubahan ijin akses home directory <user> hanya dapat dilakukan pada root.

```
$ chmod g+rwx /home/<user>contoh :chmod g+rwx /home/student
$ ls –l /home
$ exit
```

6. Sekarang cobalah untuk subtitute user ke<user-baru>(student1). Cobalahuntuk mengakses file f1

```
$ su <user-baru>
$ ls –l f1
$ cat f1
$ exit
```

7. Mengubah group dengan perintah `chgrp`

```
$ $ grep root /etc/group
$ grep other /etc/group
$ su
$ chgrp root f1
$ ls –l f1
$ chgrp <group-baru> f3
$ ls –l f3
$ exit
```

Percobaan 2 : User Mask

13. Menentukan ijin akses awal pada saat file atau direktori dibuat

```
$ touch myfile
$ ls -l myfile
```

15. Melihat nilai umask

```
$ umask
```

6. Modifikasi nilai umask

```
$ umask 027
$ umask
$ touch file_baru
$ mkdir mydir
$ ls -l
$ umask 077
$ touch xfiles
$ mkdir xdir $
ls -l
```

LATIHAN:

9. Lakukan tiga cara berbeda untuk setting ijin akses ke file atau direktori menjadi `r--r--r--`. Buatlah sebuah file dan lihat apakah yang anda lakukan benar.
10. Buatlah suatu kelompok. Copy-kan `/bin/sh` ke home directory. Ketik "`chmod +ssh`". Cek ijin akses `sh` pada daftar direktori. Sekarang tanyakan ke teman satukelompok anda untuk mengubah ke home directory anda dan menjalankan pr ogram

`./sh` dan menjalankan id command. Apa yang terjadi. Untuk keluar dari shell tekan exit.

3. Hapus `sh` dari home directory (atau setidaknya kerjakan perinta h `chmod -s sh`)
22. Modifikasi ijin akses ke home directory anda sehingga sangat privat. Cek apakah teman anda tidak dapat mengakses directory anda. Kemudian kembalikan ijin akses ke semula.
23. Ketikkan `umask 000` dan kemudian buatlah file yang bernama `world.txt` yang berisi beberapa kata "`hello world`". Lihat ijin akses pada file. Apa yang terjadi? Sekarang ketikkan `umask 022` dan buatlah file bernama `world2.txt`. Apakah perintah tersebut lebih berguna ?
24. Buatlah file yang bernama "`hello.txt`" pada home directory menggunakan perintah `cat -u > hello.txt`. Tanyakan ke teman Anda untuk masuk ke home directory Anda dan menjalankan `tail -f hello.txt`. Sekarang ketikkan beberapa baris dalam `hello.txt`. Apa yang terjadi pada layer teman Anda ?

LAPORAN RESMI:

12. Analisa hasil percobaan yang Anda lakukan.
13. Kerjakan latihan diatas dan analisa hasil tampilannya.
14. Berikan kesimpulan dari praktikum ini.

MODUL PRAKTIKUM

SISTEM OPERASI

PERTEMUAN 9

Manajemen Hardware

Universitas Esa Unggul

Jakarta – 2008

Praktikum 9

Manajemen Perangkat Keras

POKOK BAHASAN:

- ✓ Manajemen Perangkat Keras

TUJUAN BELAJAR:

Setelah mempelajari materi dalam bab ini, mahasiswa diharapkan mampu:

- ✓ Mengetahui bagaimana melihat perangkat keras yang terpasang pada sistem komputer.
- ✓ Menggunakan perintah mount dan umount pada sistem file
- ✓ Menggunakan perintah-perintah untuk manajemen perangkat keras.

DASAR TEORI:

1 FILE PERANGKAT KERAS

/dev berisi file device (perangkat) yang merupakan aspek penting pada sistem file Linux. /dev/cdrom dan /dev/fd0 merupakan drive CD-ROM dan floppy pada komputer Anda. Kita dapat melkaukan akses read dan write pada perangkat. Sebagai contoh /dev/dsp merupakan perangkat speaker. Sembarang data yang ditulis ke file ini akan dialihkan ke speaker. 'cat /boot/vmlinuz > /dev/dsp' menyebabkan kita dapat mendengarkan suara dari speaker. Untuk mencetak file dapat dikirim ke perangkat /dev/lp0. Mengirim data ke dan membaca data dari /dev/ttyS0 akan menyebabkan komunikasi dengan perangkat modem.

Mayoritas device berupa block device atau character device. Block device adalah device yang menyimpan atau membawa data, character device adalah device

yang mengirim atau transfer data. Sebagai contoh, diskette drive, hard drive dan CD-ROM drive adalah block device, seda ngkaan serial port, mouse dan paralel printer adalah character device.

Beberapa file perangkat yang umum digunakan yang perlu diingat adalah : /dev/ttyS0 (First communication port, COM1) : First serial port (mouse, modem) /dev/psaux (PS/2) : PS/2 mouse connection (mouse, keyboard)
/dev/lp0 (First printer port, LPT1) : First parallel prot (printer, scanner dsb) /dev/dsp (First audio device) : sound card, digitized voice dan PCM
/dev/usb (USB Device) : node USB device
/dev/sda (C:/SCSI device) : First SCSI de vice (HDD, Memory stick, external mass storage device seperti CD-ROM pada laptop)
/dev/scd (D:\, SCSI CD-ROM device) : First SCSI CD-ROM device
/dev/js0 (Standard gameport joystick) : First joystick device

Device didefinisikan sebagai tipe seperti block atau character dan nomor mayor dan minor. Nomor mayor digunakan untuk melakukan katagori device dan nomor minor untuk mengidentifikasi tipe device khusus. Sebagai contoh, semua IDE device dihubungkan dengan primary controller mempunyai nomor mayor 3. Perangkat master dan slave, didefinisikan lebih jauh dengan nomor minor. Terdapat dua nomor sebelum tanggal yang tercetak. Jika kita lakukan perintah ls – l /hd* maka akan terlihat nomor mayor untuk perangkat hda dan hdb adalah 3. Nomor minor berubah untuk setiap partisi tertentu. Kita dapat selalu membuat perangkat menggunakan skrip MAKEDEV dimana akan diletakkan pada directory /dev.

```
# MAKEDEV *
```

2 PERINTAH MOUNT dan UMOUNT

Sebelum menggunakan sistem file, harus di-mount terlebih dahulu. Kemudian sistem operasi dapat mengerjakan penyimpanan file. Karena semua file UNIX berada pada satu pohon direktori, operasi mount akan terlihat seperti isi dari sub direktory yang ada pada sistem file yang sudah dilakukan mounting. Contoh perintah mount

```
$ mount /dev/hda2 /home
$ mount /dev/hda3 /usr
```

Perintah mount mempunyai 2 argumen, argumen pertama adalah file device yang berhubungan dengan disk atau partisi dari sistem file. Argumen kedua adalah direktory yang dimounting. Perintah diatas berarti bahwa ”/dev/hda2 dilakukan mounting ke /home” begitu juga dengan /usr. Perbedaan antara file device /dev/hda2 dan direktory mount /home adalah file device memberikan akses ke isi disk mentah, direktory mount memberikan akses ke file dari disk. Direktory mount disebut mount point.

Linux mendukung beberapa tipe sistem file. Mount akan menebak tipe dari sistem file. Opsi –t fstype akan memberikan spesifikasi tipe sistem flie. Sebagai contoh, untuk mount floppy MS-DOS, dapat menggunakan perintah berikut :

```
$ mount –t msdos /dev/fd0 /floppy
```

Sistem file root dilakukan mounting pada waktu booting. Jika sistem file root tidak dapat dimounting, sistem tidak dapat melakukan booting. Nama sistem file dimounting sebagai root. Sistem file root mula-mula bersifat read-only. Skrip startup kemudian menjalankan fsck untuk melakukan verifikasi validitas dan jika tidak ada permasalahan, dilakukan mounting lagi sehingga write diperbolehkan. Fsck tidak boleh dijalankan pada saat sistem file dimounting, karena setiap perubahan ke sistem file saat fsck berjalan mengakibatkan kesalahan. Bila sistem file root dimounting read-only saat dilakukan pengecekan, fsck dapat memperbaiki permasalahan.

Jika sistem file tidak diperlukan untuk dimounting, dapat dilakukan unmounting dengan perintah umount. Perintah umount mempunyai satu argumen berupa file device atau mount point. Sebagai contoh untuk unmount direktory pada contoh diatas dapat digunakan perintah

```
$ umount /dev/hda2
$ umount /usr
```

Kita dapat melihat perangkat floppy da n mount point yang diijinkan pada /etc/fstab.

```
$ cat /etc/fstab
/dev/fd0   /mnt/floppy  auto    rw,user,noauto  0 0
/dev/hdc   /mnt/cdrom   iso9660 ro,user,noauto  0 0
/dev/hdc   /mnt/cdrom   iso9660 0 0 0
```

Kolom terdiri dari file device, directory mounting, tipe sistem file, opsi, frekuensi backup, fsck pass number (0 berarti tanpa cek). Opsi noauto menghentikan mounting yang dilakukan secara otomatis jika sistem dimulai (misalnya menghentikan mount –a). Opsi user mengijinkan sembarang user melakukan mounting sistem file dan karena alasan keamanan, eksekusi program tidak diijinkan (normal atau setuid)

Jika ingin menyediakan akses ke beberapa tipe floppy, perlu diberikan beberapa mount point. Setting berbeda untuk setiap mount point. Seba gai contoh untuk memberikan akses ke floppy MS-DOS dan ext2, dilakukan perubahan baris pada /etc/fstab :

```
/dev/fd0 /dosfloppy msdos user,noauto 0 0
/dev/fd0 /ext2floppy ext user,noauto 0 0
```

TUGAS PENDAHULUAN :

Sebagai tugas pendahuluan, jawablah pertanyaan-pertanyaan berikut ini :

- ü Perangkat keras diakses oleh sistem operasi Linux melalui directory /dev. Apa saja isi sub directory /dev dan sebutkan perangkatnya.
- ü Apa yang dimaksud dengan block device dan character device ? Sebutkan contoh perangkat yang merupakan block device dan character device.
- ü Apa yang dimaksud dengan mounting ? Apa maksud perintah mount dan umount ?

PERCOBAAN :

23. Pada percobaan ini setiap mahasiswa harus membawa sebuah floppy disk dan atau CDROM
24. Login sebagai user.
3. Bukalah Console Termina l dan lakukan percobaan-percobaan di bawah ini kemudian analisa hasil percobaan.
4. Selesaikan soal-soal latihan.

Percobaan 1 : Melihat perangkat pada sistem komputer

13. Melihat daftar perangkat. Perhatikan apakah perangkat-perangkat yang disebutkan pada dasar teori terdapat pada komputer anda. Perhatikan tipe perangkat berupa block device atau character device. Apa yang membedakan suatu perangkat merupakan block device atau character device?

```
$ ls -l /dev
```

14. Perhatikan nomor mayor dan minor pada perangkat hard disk Anda. Apa maksudnya ?

```
$ ls -l /dev/hd*
```

Percobaan 2 : Menangani Removable Media

24. Melihat daftar perangkat yang ada pada sistem file utama. Perhatikan titik mount untuk perangkat floppy dan CDROM. Perhatikan opsi yang ada jelaskan maksudnya.

```
$ cat /etc/fstab
```

25. Cobalah melakukan mounting pada floppy disk

```
$ mount /dev/fd0 /mnt/floppy
$ cd /mnt/floppy
$ ls -l
```

3. Agar semua perubahan data tertulis pada floppy dan mengambil floppy disk sistem file gunakan perintah umount.

```
$ cd
$ umount /mnt/floppy
```

4. Lakukan hal yang sama untuk perangkat CDROM.

Percobaan 3 : Melak ukan format MSDOS pada floppy

15. Linux dapat membaca dan menulis dengan format MSDOS maupun Linux. Untuk menggunakan floppy MS, dapat digunakan perintah MS -DOS dengan didahului huruf ”m”. Misalnya, ”mdir a:” akan melihat daftar file pada drive a, ”mcopy” melakukan copy file, ”mdel” melakukan penghapusan file. Lakukan format floppy dengan perintah

```
$ fdformat /dev/fd0H1440
$ mformat a:
```

2. Cobalah melakukan list directory, copy dan delete file

```
$ mdir a:
$ mcopy <namafile> a:
$ mdel a:/<namafile>
```

14. Lakukan pembuatan direktory pada floppy dengan perintah mmd, copy file dengan mcopy, delete file dengan mdel, pindah directory dengan mcd dan melihat isi directory dengan mdir.
15. Lakukan format floppy disk menggunakan perintah mkfs

```
$ mkfs –t msdos /dev/fd0
```

5. Sebelum menggunakan floppy yang sudah terformat la kukan mounting sistem file

```
$ mount /mnt/floppy
```

10. Untuk melihat apakah floppy sedang digunakan ketikkan `$ df`
11. Lakukan unmount terhadap floppy disk.

```
$ umount /mnt/floppy
```

LATIHAN:

13. Lihatlah directory /proc/devices yang berisi perangkat-perangkat yang terdapat pada sistem komputer. Perlihatkan tampilannya dan sebutkan block device dan character device apa saja yang terdapat pada sistem komputer.
14. Lakukan operasi file dan directory dengan menggunakan perintah MS-DOS seperti mdir, mmd, mcd, mcopy dan mdel, mmove . Tuliskan perintah yang anda lakukan.
15. Lakukan mounting terhadap floppy disk kemudian cobalah pindah ke directory /mnt/floppy dan lakukan operasi file dan directory (perintah cp, rm, mkdir, rmdir, cd, move).
14. Lihat manual dari fdisk dan fsck, kemudian lakukan percobaan menggunakan perintah tersebut.
15. Lihat manual dari perintah mke2fs, kemudian lakukan percoba an dengan menggunakan perintah tersebut.

LAPORAN RESMI:

16. Analisa hasil percobaan yang Anda lakukan.
17. Kerjakan latihan diatas dan analisa hasil tampilannya.
18. Berikan kesimpulan dari praktikum ini.

MODUL PRAKTIKUM

SISTEM OPERASI

PERTEMUAN 10

Linux Booting Process

Universitas Esa Unggul

Jakarta – 2008

Praktikum10

Linux Booting Process

POKOK BAHASAN:

- ✓ Linux BootingProcess

TUJUAN BELAJAR:

Setelah mempelajari materi dalam bab ini, mahasiswa diharapkan mampu:

- ✓ Mengetahui inisialisasi booting proses pada sistem operasi Linux
- ✓ Melakukan perubahan inisia lisasi bootingproses

DASAR TEORI:

1 PCBOOTDANLINUXINITPROCESS

Pada praktikum ini membahas PC boot process dan inisialisasi sistem operasi Linux pada aplikasi background (daemons/service).

1. BIOS : Basic Input/Output System adalah antar muka level terendah antara komputer dan peripheral. Bios melakukan pemeriksaan pada memori dan mencari instruksipadaMasterBootRecord(MBR)padafloppyatauharddrive.
2. MBR menunjuk ke boot loader (LILO : Linux bootloader)
3. LILO akan menanyakan label sistem operasi yang akan mengidentifikasi kernel yang dijalankan. Kernel akan menjalankan sistem operasiLinux.
4. Yang pertama kali dikerjakan oleh kernel adalah menjalankan program init. Init adalahroot/parentdarisemuaprosesyangdijalankanpadaLinux

5. Proses per tama yang memulai init adalah skrip`/etc/rc.d/rc/sysinit`.

1
2
6

6. Berdasarkan run-level yang ditentukan, skrip dieksekusi untuk memulai proses tertentuuntukmenjalankansistemdanmembuatsistemlebihfungsional.

2. LINUX INIT PROCESS

Proses init adalah langkah terakhir pada prosedur boot dan diidentifikasi sebagai processide"1". Init bertanggung-jawab untuk memulai proses sistem seperti yangditentukan pada file `/etc/inittab`. Init biasanya memulai "`getty`" yang menunggu layar login yang menandakan proses shellseorang user. Pada saat shutdown, initmengontrol urutan dan prosesuntushutdown. Proses init tidak pernahshutdown.Proses init merupakan proses user dan bukan proses sistem kernel meskipun dijalankan sebagairoot.

Proses sistem :

ProcessID	Description
0	TheScheduler
1	The initprocess
2	kflushd
3	kupdate
4	kpiod
5	kswapd
6	mdrecoveryd

3. PROSEDUR BOOT

Linux mempunyai 6 state operasi dimana "0" adalah shutdown state dan "3" keatas adalah operasional penuh dengan semua proses yang esensial dijalankan untuk interaksi user. Berdasarkan sistem boot, Linux sistem akan melakukan :

- Mengeksekusi program `/sbin/init` yang memulai semua proses-proses lain. Program ini akan diberikan ke mesin oleh proses awal yang didefinisikan padafile `/etc/inittab`
- Komputerakandi-bootingkerunlevelyangdidefinisikanolehbaris`initdefault` pada file `/etc/in`