Modul Praktikum

Sistem Operasi Linux

PRAKTIKUM 1
Perintah Dasar Sistem Operasi Linux

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
```

Percobaan 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 directory

   ```
   $ ls
   ```

2. Melihat semua file lengkap

   ```
   $ ls -l
   ```

3. Menampilkan semua file atau direktori yang tersembunyi

   ```
   $ ls -a
   ```

4. Menampilkan semua file atau direktori tanpa proses sorting

   ```
   $ ls -f
   ```

5. Menampilkan isi suatu direktori

   ```
   $ ls /usr
   ```

6. Menampilkan isi direktori root

   ```
   $ ls /
   ```

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

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

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

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

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

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

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.

PRAKTIKUM 2
Operasi Input/Output (I/O)

1. Login sebagai user.
2. Bukalah Console Terminal dan lakukan percobaan-percobaan di bawah ini. Perhatikan hasil setiap percobaan.
3. 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

```
$ mail very@ftik.usm.ac.id
contoh 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.txt
Badu
Zulkif
li
Yulizi
r Yudi
Ade
[Ctrl-d]
$ cat > kelas2.txt
Budi
Gama
Asep
Muchli
s
[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 input.
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.txt dog
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.

PRAKTIKUM 3
Operasi File & Struktur Direktori

1. Login sebagai user.
2. Bukalah Console Terminal dan lakukan percobaan-percobaan di bawah ini. Perhatikan hasilnya.
3. 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. Menghapus satu atau lebih direktori hanya dapat dilakukan pada direktori kosong dan hanya dapat dihapus oleh pemiliknya kecuali bila diberikan ijin aksesnya

```
$ rmdir B          (Terdapat pesan error, mengapa ?)
$ ls -l B
$ rmdir B/F B
$ ls -l B          (Terdapat pesan error, mengapa ?)
```

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 > contoh
Membuat 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*

1. 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 '{}' ';'
```

2. Perintah which

```
$ which ls
```

3. 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.
3. Telusuri direktory /dev. Identifikasi perangkat yang tersedia. Identifikasi tty (terminal) Anda (ketik whoami); siapa pemilih tty Anda (gunakan ls –l).
4. 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 ?
5. Ubahlah direktory home ke user lain secara langsung menggunakan cd ~username.
6. Ubah kembali ke direktory home Anda.
7. Buat subdirektory work dan play.
8. Hapus subdirektory work.
9. Copy file /etc/passwd ke direktory home Anda.
10. Pindahkan ke subirectory play.
11. 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 ?
12. Buatlah file bernama hello.txt yang berisi kata ”hello word”. Dapatkah Anda gunakan ”cp” menggunakan ”terminal” sebagai file asal untuk menghasilkan efek yang sama ?
13. Copy hello.txt ke terminal. Apa yang terjadi ?
14. Masih direktory home, copy keseluruhan direktory play ke direktory bernama work menggunakan symbolic link.
15. Hapus direktory work dan isinya dengan satu perintah.

LAPORAN RESMI:

1. Analisa hasil percobaan yang Anda lakukan.
   a. Analisa setiap hasil tampilannya.
   b. Pada Percobaan 1 point 3 buatlah pohon dari struktur file dan direktori
   c. Bila terdapat pesan error, jelaskan penyebabnya.
2. Kerjakan latihan diatas dan analisa hasil tampilannya.
3. Berikan kesimpulan dari praktikum ini.

PRAKTIKUM 4
Proses & Manajemen Proses

1. Login sebagai user.
2. Lakukan percobaan-percobaan di bawah ini kemudian analisa hasil percobaan.
3. Selesaikan soal-soal latihan.

Percobaan 1 : Status Proses

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

2. 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
   ```

3. 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
   ```

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

   ```
   $ ps –u <user>
   ```

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

   ```
   $ ps –a
   $ ps –au
   ```

6. **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.

2. 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
   ```

3. 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
   ```

4. 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` mempunyai proses `bash` sebagai child. Proses `bash` mempunyai proses child `startx`. Proses `startx` mempunyai child `xinit` dan seterusnya.

   ```
   $ pstree
   ```

5. Ketik **pstree | grep mingetty** dan tekan **Enter**. Akan menampilkan semua proses `mingetty` yang berjalan pada system yang berupa *console virtual*. Selain menampilkan semua proses, proses dikelompokkan dalam satu baris dengan suatu angka sebagai jumlah proses yang berjalan.

   ```
   $ pstree | grep mingetty
   ```

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

   ```
   $ pstree –p
   ```

7. Untuk menampilkan proses dan ancestor yang tercetak tebal gunakan opsi **–h**.

   ```
   $ pstree –h
   ```

Percobaan 3 : Menampilkan Status Proses dengan Berbagai Format

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

2. 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.

3. 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.

4. Ketik **ps –ef | more** dan tekan **Enter**. Opsi **–ef** 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.

5. 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 terlihat prompt `--More--` di bagian bawah screen, tekan **q** untuk kembali ke prompt perintah.

6. 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 ditampilkan 0.

   ```
   $ ps –eo pid,ppid,%mem,cmd | more
   ```

7. **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
   ```

6. 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
   ```

Percobaan 5 : Menghentikan dan memulai kembali job

1. Cara lain meletakkan job pada *background* dengan memulai job secara normal (pada *foreground*), **stop** job dan memulai lagi pada *background*

   ```
   $ yes > /dev/null
   ```

   Hentikan sementara job (*suspend*), bukan menghentikannya (*terminate*), tetapi menghentikan sementara job sampai di restart. Untuk menghentikan sementara job gunakan **Ctrl-Z**.

2. Untuk restart job pada *foreground*, gunakan perintah `fg`.

   ```
   $ fg
   ```

3. Shell akan menampilkan nama perintah yang diletakkan di *foreground*. Stop job lagi dengan **Ctrl-Z**. Kemudian gunakan perintah `bg` untuk meletakkan job pada *background*.

```
$ bg
```

4. Job tidak bisa dihentikan dengan **Ctrl-Z** karena job berada pada *background*. Untuk menghentikannya, letakkan job pada *foreground* dengan `fg` dan kemudian hentikan sementara dengan **Ctrl-Z**.

   ```
   $ fg
   ```

5. Job pada *background* dapat digunakan untuk menampilkan teks pada terminal, dimana dapat diabaikan jika mencoba mengerjakan job lain.

   ```
   $ yes &
   ```

   Untuk menghentikannya tidak dapat menggunakan **Ctrl-C.** Job harus dipindah ke *foreground*, baru dihentikan dengan cara tekan **fg** dan tekan **Enter**, kemudian dilanjutkan dengan **Ctrl-Z** untuk menghentikan sementara.

6. Apabila ingin menjalankan banyak job dalam satu waktu, letakkan job pada *foreground* atau *background* dengan memberikan job ID

   ```
   $ fg %2      atau   $ %2
   $ bg %2
   ```

   Tekan **fg** dan tekan **Enter**, kemudian dilanjutkan dengan **Ctrl-Z** untuk menghentikan sementara.

7. Lihat job dengan perintah **ps -fae** dan tekan **Enter**. Kemudian hentikan proses dengan perintah kill.

   ```
   $ ps -fae
   $ kill -9 <NomorPID>
   ```

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

Percobaan 6 : Percobaan dengan Penjadwalan Prioritas

1. Login sebagai root.

2. Buka 3 terminal, tampilkan pada screen yang sama.

3. Pada setiap terminal, ketik **PS1 = "\w:"** diikuti **Enter**. **\w** menampilkan path pada direktori home.

4. Karena login sebagai root, maka akan ditampilkan **~:** pada setiap terminal. Untuk setiap terminal ketik **pwd** dan tekan **Enter** untuk melihat bahwa Anda sedang berada pada direktori `/root`.

5. Buka terminal lagi (keempat), atur posisi sehingga keempat terminal terlihat pada screen.

6. Pada terminal keempat, ketik **top** dan tekan **Enter.** Maka program `top` akan muncul. Ketik **i**. **Top** akan menampilkan proses yang aktif. Ketik **lmt**. **Top** tidak lagi menampilkan informasi pada bagian atas dari screen. Pada percobaan ini, terminal ke empat sebagai jendela **Top**.

7. Pada terminal 1, bukalah program executable C++ dengan mengetik program **yes** dan tekan **Enter.**

8. Ulangi langkah 7 untuk terminal 2.

9. Jendela **Top** akan menampilkan dua program `yes` sebagai proses yang berjalan. Nilai %CPU sama pada keduanya. Hal ini berarti kedua proses mengkonsumsi waktu proses yang sama dan berjalan sama cepat. PID dari kedua proses akan berbeda, misalnya 3148 dan 3149. Kemudian gunakan terminal 3 (yang tidak menjalankan `primes` maupun Jendela **Top**) dan ketik **renice 19 <PID terminal 1>** (contoh : **renice 19 3148**) dan diikuti **Enter**. Hal ini berarti mengganti penjadwalan prioritas dari proses ke 19.

10. Tunggu beberapa saat sampai program `top` berubah dan terlihat pada jendela **Top**. Pada kolom STAT memperlihatkan N untuk proses 3148. Hal ini berarti bahwa penjadwalan prioritas untuk proses 3148 lebih besar (lebih lambat) dari 0. Proses 3149 berjalan lebih cepat.

11. Program `top` juga mempunyai fungsi yang sama dengan program `renice`. Pilih Jendela **Top** dan tekan **r**. Program `top` terdapat prompt **PID to renice:** tekan **3148** (ingat bahwa Anda harus mengganti 3148 dengan PID Anda sendiri) dan tekan **Enter**. Program `top` memberikan prompt **Renice PID 3148 to value:** tekan **-19** dan tekan **Enter**.

12. Tunggu beberapa saat sampai `top` berubah dan lihat nilai %CPU pada kedua proses. Sekarang proses 3148 lebih cepat dari proses 3149. Kolom status menunjukkan < pada proses 3148 yang menunjukkan penjadwalan prioritas lebih rendah (lebih cepat) dari nilai 0.

13. Pilih terminal 3 (yang sedang tidak menjalankan `yes` atau program `top`) dan ketik **nice –n -10 yes** dan tekan **Enter.** Tunggu beberapa saat agar program `top` berubah dan akan terlihat proses `primes` ketiga. Misalnya PID nya 4107. Opsi -10 berada pada kolom NI (penjadwalan prioritas).

14. Jangan menggunakan mouse dan keyboard selama 10 detik. Program `top` menampilkan proses yang aktif selain program `yes`. Maka akan terlihat proses `top` terdaftar tetapi %CPU kecil (dibawah 1.0) dan konsisten. Juga terlihat proses berhubungan dengan dekstop grafis seperti X, panel dll.

15. Pindahkan mouse sehingga kursor berubah pada screen dan lihat apa yang terjadi dengan tampilan `top`. Proses tambahan akan muncul dan nilai %CPU berubah sebagai bagian grafis yang bekerja. Satu alasan adalah bahwa proses 4107 berjalan pada penjadwalan prioritas tinggi. Pilih jendela **Top**, ketik **r**. **PID to renice**: muncul prompt. Ketik **4107** (ubahlah 4107 dengan PID Anda) dan tekan **Enter**. **Renice PID 4107 to value:** muncul prompt. Ketik **0** dan tekan **Enter**. Sekarang pindahkan mouse ke sekeliling screen. Lihat perubahannya.

16. Tutup semua terminal window.

17. Logout dan login kembali sebagai user.

LATIHAN:

1. Masuk ke tty2 dengan **Ctrl+Alt+F2**. Ketik **ps –au** dan tekan **Enter.** Kemudian perhatikan keluaran sebagai berikut :
   a. Sebutkan nama-nama proses yang bukan root.
   b. Tulis PID dan COMMAND dari proses yang paling banyak menggunakan CPU time.
   c. Sebutkan buyut proses dan PID dari proses tersebut
   d. Sebutkan beberapa proses daemon
   e. Pada prompt login lakukan hal-hal sebagai berikut :

```
$ csh
$ who
$ bash
$ ls
$ sh
$ ps
```

   f. Sebutkan PID yang paling besar dan kemudian buat urut-urutan proses sampai ke PPID = 1.

2. Cobalah format tampilan ps dengan opsi berikut dan perhatikan hasil tampilannya :

   - **-f** daftar penuh
   - **-j** format job
   - **j** format job control
   - **l** daftar memanjang
   - **s** format sinyal
   - **v** format virtual memory
   - **X** format register i386

3. Lakukan urutan pekerjaan berikut :

   a. Gunakan perintah find ke seluruh direktory pada sistem, belokkan output sehingga daftar direktori dialihkan ke file directories.txt dan daftar pesan error dialihkan ke file errors.txt

   b. Gunakan perintah `sleep 5`. Apa yang terjadi dengan perintah ini ?

   c. Jalankan perintah pada *background* menggunakan &

   d. Jalankan `sleep 15` pada *foreground*, hentikan sementara dengan Ctrl-Z dan kemudian letakkan pada *background* dengan `bg`. Ketikkan `jobs`. Ketikkan `ps`. Kembalikan job ke *foreground* dengan perintah `fg`.

   e. Jalankan `sleep 15` pada *background* menggunakan & dan kemudian gunakan perintah `kill` untuk menghentikan proses diikuti *job number*.

   f. Jalankan `sleep 15` pada *background* menggunakan & dan kemudian gunakan `kill` untuk menghentikan sementara proses. Gunakan `bg` untuk melanjutkan menjalankan proses.

   g. Jalankan `sleep 60` pada *background* 5 kali dan terminasi semua pada dengan menggunakan perintah `killall`.

   h. Gunakan perintah `ps`, `w` dan `top` untuk menunjukkan semua proses yang sedang dieksekusi.

   i. Gunakan perintah `ps -aeH` untuk menampilkan hierarki proses. Carilah init proses. Apakah Anda bisa identifikasi sistem daemon yang penting ? Dapatkan Anda identifikasi shell dan subproses ?

   j. Kombinasikan `ps -fae` dan `grep`, apa yang Anda lihat?

   k. Jalankan proses `sleep 300` pada *background*. Log off komputer dan log in kembali. Lihat daftar semua proses yang berjalan. Apa yang terjadi pada proses sleep ?

LAPORAN RESMI:

1. Analisa hasil percobaan yang Anda lakukan.
2. Kerjakan latihan diatas dan analisa hasil tampilannya.
3. Berikan kesimpulan dari praktikum ini.

PRAKTIKUM 5
Sistem File – Atribut & Ijin Akses

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

Percobaan 1 : Ijin Akses

1. 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> f1 contoh : chown student1 f1
$ ls -l f1
```

5. 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
```

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

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

6. 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

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

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

2. Melihat nilai umask

```
$ umask
```

3. Modifikasi nilai umask

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

LATIHAN:

1. 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.
2. Buatlah suatu kelompok. Copy-kan `/bin/sh` ke home directory. Ketik "`chmod +s sh`". Cek ijin akses `sh` pada daftar direktori. Sekarang tanyakan ke teman satu kelompok anda untuk mengubah ke home directory anda dan menjalankan program `./sh` dan menjalankan id command. Apa yang terjadi ?. Untuk keluar dari shell tekan exit.
3. Hapus `sh` dari home directory (atau setidaknya kerjakan perintah `chmod -s sh`).
4. 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.
5. 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 ?
6. 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:

1. Analisa hasil percobaan yang Anda lakukan.
2. Kerjakan latihan diatas dan analisa hasil tampilannya.
3. Berikan kesimpulan dari praktikum ini.

PRAKTIKUM 6
Manajemen Perangkat Keras

1. Pada percobaan ini setiap mahasiswa harus membawa sebuah floppy disk dan atau CDROM
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 : Melihat perangkat pada sistem komputer

1. 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
   ```

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

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

Percobaan 2 : Menangani Removable Media

1. 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
   ```

2. 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 : Melakukan format MSDOS pada floppy

1. 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>
   ```

3. 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.

4. Lakukan format floppy disk menggunakan perintah mkfs

   ```
   $ mkfs –t msdos /dev/fd0
   ```

5. Sebelum menggunakan floppy yang sudah terformat lakukan mounting sistem file

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

6. Untuk melihat apakah floppy sedang digunakan ketikkan

   ```
   $ df
   ```

7. Lakukan unmount terhadap floppy disk.

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

LATIHAN:

1. 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.
2. Lakukan operasi file dan directory dengan menggunakan perintah MS-DOS seperti mdir, mmd, mcd, mcopy dan mdel, mmove. Tuliskan perintah yang anda lakukan.
3. 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).

4. Lihat manual dari fdisk dan fsck, kemudian lakukan percobaan menggunakan perintah tersebut.
5. Lihat manual dari perintah mke2fs, kemudian lakukan percobaan dengan menggunakan perintah tersebut.

PRAKTIKUM 7
Manajemen User & Group

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

Percobaan 1 : Melihat file `/etc/passwd` dan `/etc/group`

1. Lihatlah isi file `/etc/passwd` dan sebutkan kolom apa saja yang terdapat pada setiap baris.

```
# cat /etc/passwd | more
```

2. Lihatlah isi file `/etc/group` dan sebutkan kolom apa saja yang terdapat pada setiap baris.

```
# cat /etc/group | more
```

Percobaan 2 : Menambah group user

1. Buatlah 3 group user baru dengan perintah `groupadd`. Perhatikan informasi group user baru pada file `/etc/group`.

```
# groupadd friend
# groupadd classmate
# groupadd neighbour
# cat /etc/group
```

Percobaan 3 : Menambah User

1. Buatlah user baru dengan perintah useradd. Perhatikan perubahan isi file /etc/passwd setelah pembuatan user baru. Juga perhatikan apakah home direktory setiap user juga dibuat pada saat pembuatan user baru.

```
# useradd –g friend bob
# grep bob /etc/passwd
# useradd lili
# passwd lili
# grep lili /etc/passwd
# ls –l /home
```

2. Opsi –g pada perintah useradd untuk menentukan group dari user yang dibuat.

```
# useradd –g neighbour jane
# ls –l /home
```

Percobaan 4 : Memodifikasi group dari user

1. Dengan perintah usermod, modifikasi group dari Setiap user merupakan memilih suatu group primer dan kemungkinan juga bagian dari group lain (supplementary group). Untuk memodifikasi group dari suatu user dapat digunakan perintah usermod.

```
# usermod -g classmate -G friend,neighbour bob
# usermod -g friend -G classmate lili
```

Percobaan 5 : Melihat group dari user

1. Lihat group dari seorang user dengan perintah groups.

```
# groups bob
# groups lili
# groups jane
```

Percobaan 6 : Mengubah password user

1. Root dapat mengubah password dari user.

```
# passwd bob
```

2. Password yang diubah dengan perintah `usermod` merupakan file enkripsi, sehingga tidak dapat digunakan sebagai password pada saat login.

```
# useradd -g friend diane
# usermod -p diane diane
```

3. Cobalah login sebagai diane, apakah anda dapat login ?

4. Cobalah mengubah password user dengan login pada user yang bersangkutan. Login sebagai user, dan ubahlah password user.

```
$ passwd
```

Percobaan 7 : Menghapus user

1. Hapus user dengan menggunakan perintah userdel. Opsi –r untuk menghapus seluruh isi home directory.

```
# userdel -r bob
# userdel -r lili
# userdel -r jane
# userdel -r diane
```

Percobaan 8 : Menghapus group

1. Hapus group dengan menggunakan perintah userdel.

```
# groupdel friend
# groupdel classmate
# groupdel neighbour
```

Percobaan 9 : Menghapus home directory

1. Hapus home direktory.

```
# rmdir /home/bob
# rmdir /home/lili
# rmdir /home/jane
# rmdir /home/diane
```

LATIHAN:

1. Buatlah tiga group "parent", "children" dan "soho". Perhatikan anggota dari setiap grup berikut :

Parents	Children	Soho
Paul	Alice	Accounts
Jane	Derek	Sales

2. Buatlah user account untuk setiap anggota group sesuai tabel diatas.
3. Cek apakah home direktory yang terbentuk sesuai dengan tabel diatas.
4. Ubahlah password Paul dan Derek melalui root.
5. Cobalah mengubah password Alice dengan login sebagai Alice
6. Lihat keanggotaan dari setiap user.
7. Hapuslah user Account dan Sales.

LAPORAN RESMI:

1. Analisa hasil percobaan yang Anda lakukan.
2. Kerjakan latihan diatas dan analisa hasil tampilannya.
3. Berikan kesimpulan dari praktikum ini.

PRAKTIKUM 8
Pemrograman Shell

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

Percobaan 1 : Membuat shell script

1. Buatlah file prog01.sh dengan editor vi

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

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

```
$ . prog01.sh
```

3. 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
```

2. 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
```

3. Menggabungkan dua variable atau lebih

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

4. Menggabungkan isi variable dengan string yang lain. Jika digabungkan dengan nama variable yang 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
```

6. Modifikasi file `prog01.sh` berikut

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

7. 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 nama
amir
$ echo $nama
```

2. Membaca nama dan alamat dari keyboard

```
$ vi prog02.sh
#!/bin/sh
# prog02.sh
# membaca nama dan alamat
echo "Nama Anda : "
read nama
echo "Alamat : "
read alamat
echo "Kota : "
read kota
echo
echo "Hasil adalah : $nama, $alamat di $kota"
```

3. Eksekusi program prog02.sh

```
$ . prog02.sh
Nama Anda :
Amir
Alamat :
Jl semangka 67
Kota :
Surabaya
Hasil adalah : Amir, Jl semangka di Surabaya
```

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

```
$ 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
```

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

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

8. Buatlah shell script prog03.sh

```
$ vi prog03.sh
#!/bin/sh
# prog03.sh
#
NAMA=`whoami`
echo Nama Pengguna Aktif adalah $NAMA
tanggal=`date | cut -c1-10`
echo Hari ini tanggal $tanggal
```

9. Eksekusi prog03.sh

```
$ . prog03.sh
```

Percobaan 4 : Parameter

1. Membuat shell script prog04.sh

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

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

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

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 $?
```

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

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

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

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

Percobaan 7 : Konstruksi if then else

1. Membuat shell script prog05.sh

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

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

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

Percobaan 8 : Instruksi Test

1. Menggunakan instruksi test, perhatikan spasi antara

```
$ NAMA=amir
$ test $NAMA = amir
$ echo $?
$ test $NAMA = boris
$ echo $?
```

2. Aplikasi test dengan konstruksi if

```
$ vi prog06.sh
#!/bin/sh
# prog06.sh
echo –n "NAMA = "
read NAMA
if test "$NAMA" = amir
then
```

```
echo "Selamat Datang $NAMA"
else
echo "Anda bukan amir, sorry!"
fi
```

3. Jalankan program `prog06.sh` dengan memasukkan NAMA = amir dan NAMA = <CR> perhatikan hasil tampilannya

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

4. Modifikasi `prog06.sh` dengan menggunakan notasi untuk test

```
$ vi prog06.sh
#!/bin/sh
# prog06.sh
echo -n "NAMA = "
read NAMA
if [ "$NAMA" = amir ]
then
        echo "Selamat Datang $NAMA"
else
        echo "Anda bukan amir, sorry!"
fi
```

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

```
$ . prog06.sh [NAMA = amir]
```

Percobaan 9 : Notasi && dan ||

1. Bila file `prog01.sh` ada (TRUE), maka jalankan program berikutnya. File `prog01.sh` ada, karena itu exit status adalah TRUE, hasil operasi AND masih tergantung pada hasil eksekusi instruksi ke 2, dan dengan demikian instruksi echo akan dijalankan.

```
$ [ -f prog01.sh ] && echo "Prog01.sh ada"
```

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

```
$ [ -f prog99.sh ] && echo "Prog99.sh ada"
```

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

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

4. Bila `prog01.sh` ada maka jalankan program berikutnya. File `prog01.sh` memang ada, karena itu exit status adalah TRUE, dan karena sudah TRUE maka instruksi echo tidak lagi dijalankan

```
$ [ -f prog01.sh ] || echo "Dieksekusi tidak ?"
```

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

```
$ [ -f prog99.sh ] || echo "Dieksekusi tidak ?"
```

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

```
$ [ -f prog99.sh ] || echo "Sorry, prog99.sh tidak ada"
```

Percobaan 10 : Operator bilangan bulat untuk test

1. Menggunakan operator dengan notasi test

```
$ i=5
$ test "$i" -eq 5
$ echo $?
```

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

```
$ [ "$i" -eq 5 ]
$ echo $?
```

Percobaan 11 : Operator Logical dan konstruksi elif

1. Buatlah file `prog07.sh`

```
$ vi prog07.sh
#!/bin/sh
# prog07.sh
echo -n "INCOME = "
read INCOME
if [ $INCOME -ge 0 -a $INCOME -le 10000 ]
then
        BIAYA=10
elif [ $INCOME -gt 10000 -a $INCOME -le 25000 ]
then
```

```
        BIAYA=25
else
        BIAYA=35
fi
echo "Biaya = $BIAYA"
```

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

```
$ . prog07.sh [INCOME=5000]
$ . prog07.sh [INCOME=20000]
$ . prog07.sh [INCOME=28000]
```

Percobaan 12 : Hitungan aritmetika

1. Menggunakan utilitas expr

```
$ expr 5 + 1
$ A=5
$ expr $A + 2
$ expr $A – 4
$ expr $A * 2 (Ada Pesan Error)
$ expr $A \* 2
$ expr $A / 6 +10
$ expr 17 % 5
```

2. Substitusi isi variable dengan hasil utilitas expr

```
$ A=5
$ B=`expr $A + 1`
$ echo $B
```

Percobaan 13 : Instruksi exit

1. Buat shell script `prog08.sh`

```
$ vi prog08.sh
#!/bin/sh
if [ -f prog01.sh ]
then
        exit 3
else
        exit -1
fi
```

2. Jalankan script `prog08.sh` dan periksa status exit

```
$ . prog08.sh
$ echo $?
```

Percobaan 14 : Konstruksi case – esac

1. Buatlah file `prog09.sh` dengan editor vi

```
$ vi prog09.sh
#!/bin/sh
# Prog: prog09.sh
echo "1. Siapa yang aktif"
echo "2. Tanggal hari ini"
echo "3. Kalender bulan ini"
echo -n " Pilihan : "
read PILIH
case $PILIH in
1)
       echo "Yang aktif saat ini"
       who
       ;;
2)
       echo "Tanggal hari ini"
       date
       ;;
3)
       echo "Kalender bulan ini"
       cal
       ;;
*)
       echo "Salah pilih !!"
       ;;
Esac
```

2. Jalankan program prog09.sh, cobalah beberapa kali dengan inputan yang berbeda

```
$ . prog09.sh
```

3. Buatlah file prog10.sh yang merupakan bentuk lain dari case

```
$ vi prog10.sh
#!/bin/sh
# Prog: prog10.sh
echo -n "Jawab (Y/T) : "
read JWB
```

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

4. Jalankan program prog10.sh, cobalah beberapa kali dengan inputan yang berbeda

```
$ . prog10.sh
```

5. Modifikasi file prog10.sh yang merupakan bentuk lain dari case

```
$ vi prog10.sh
#!/bin/sh
# Prog: prog10.sh
echo –n "Jawab (Y/T) : \c"
read JWB
case $JWB in
  [yY] | [yY][aA] ) JWB=y ;;
  [tT] | [tT]idak ) JWB=t ;;
*) JWB=? ;;
Esac
```

6. Jalankan program prog10.sh, cobalah beberapa kali dengan inputan yang berbeda

```
$ . prog10.sh
```

Percobaan 15 : Konstruksi for-do-done

1. Buatlah file `prog11.sh`

```
$ vi prog11.sh
#!/bin/sh
# Prog: prog11.sh
for NAMA in bambang harry kadir amir
do
       echo "Nama adalah : $NAMA"
done
```

2. Jalankan program prog11.sh

```
$ . prog11.sh
```

3. Buatlah file `prog12.sh` yang berisi konstruksi for dan wildcard, program ini akan menampilkan nama file yang berada di current direktori

```
$ vi prog12.sh
```

```
#!/bin/sh
# Prog: prog12.sh
for F in *
do
        echo $F
done
```

4. Jalankan program `prog12.sh`

```
$ . prog12.sh
```

5. Modifikasi file `prog12.sh`, program ini akan menampilkan long list dari file yang mempunyai ekstensi lst

```
$ vi prog12.sh
#!/bin/sh
# Prog: prog12.sh
for F in *.lst
do
        ls -l $F
done
```

6. Jalankan program prog12.sh

```
$ . prog12.sh
```

Percobaan 16 : Konstruksi while-do-done

1. Buatlah file `prog13.sh`

```
$ vi prog13.sh
#!/bin/sh
# Prog: prog13.sh
PILIH=1
while [ $PILIH -ne 4 ]
do
        echo "1. Siapa yang aktif"
        echo "2. Tanggal hari ini"
        echo "3. Kalender bulan ini"
        echo "4. Keluar"
        echo " Pilihan : \c"
        read PILIH
        if [ $PILIH -eq 4 ]
        then
             break
        fi
```

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

2. Jalankan program `prog13.sh`

```
$ . prog13.sh
```

Percobaan 17 : Instruksi dummy

1. Modifikasi file `prog13.sh`

```
$ vi prog13.sh
#!/bin/sh
# Prog: prog13.sh

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

2. Jalankan program `prog13.sh`

```
$ . prog13.sh
```

3. Buatlah file `prog14.sh` yang berisi instruksi dummy untuk konstruksi if

```
$ vi prog14.sh
#!/bin/sh
# Prog: prog14.sh
echo -n "Masukkan nilai : "
read A
if [ $A -gt 100 ]
then
```

```
        :
else
        echo "OK !"
fi
```

4. Jalankan program `prog14.sh` beberapa kali dengan input yang berbeda

```
$ . prog14.sh
```

Percobaan 18 : Fungsi

1. Buatlah file `fungsi.sh`

```
$ vi fungsi.sh
#!/bin/sh
# Prog: fungsi.sh
F1( ) {
  echo "Fungsi F1"
  return 1
}
echo "Menggunakan Fungsi"
F1
F1
echo $?
```

2. Jalankan program `fungsi.sh`

```
$ . fungsi.sh
```

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

```
$ vi fungsi.sh
#!/bin/sh
# Prog: fungsi.sh
F1( )
{
  Honor=10000
  echo "Fungsi F1"
  return 1
}
echo "Menggunakan Fungsi"
F1
F1
echo "Nilai balik adalah $?"
echo "Honor = $Honor"
```

4. Jalankan program `fungsi.sh`

```
$ . fungsi.sh
```

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

```
$ vi fungsi.sh
#!/bin/sh
# Prog: fungsi.sh
F1( )
{
  local Honor=10000
  echo "Fungsi F1"
  return 1
}
echo "Menggunakan Fungsi"
F1
F1
echo "Nilai balik adalah $?"
echo "Honor = $Honor"
```

6. Jalankan program `fungsi.sh`

```
$ . fungsi.sh
```

LATIHAN:

1. Buatlah program ***salin.sh*** yang menyalin file (copy) sebagai berikut : *salin.sh* file-asal file-tujuan

   Dengan ketentuan :
   - Bila file asal tidak ada, berikan pesan, salin gagal.
   - Bila file tujuan ada dan file tersebut adalah directory, beri pesan bahwa file tidak bisa disalin ke direktori
   - Bila file tujuan ada dan file biasa, beri pesan apakan file tersebut akan dihapus, bila dijawab dengan "Y", maka copy file tersebut
   - Bila file tujuan belum ada, lakukan copy

   Untuk mengambil nama file, gunakan parameter $1 dan $2. Bila jumlah parameter tidak sama ($#) dengan 2, maka beri pesan exit = -1

```
#!/bin/sh
# file: salin.sh
# Usage: salin.sh fasal ftujuan
if [ $# -ne 2]
```

```
then
        echo "Error, usage: salin.sh file-asal file-tujuan"
        exit -1
fi
fasal=$1
ftujuan=$2
echo "salin.sh $fasal $ftujuan"
……
……
```

2. Buat program yang memeriksa nama direktori, jika parameter tersebut adalah direktori, maka jalankan instruksi ls –ld pada direktori tersebut. Namakan program tersebut ***checkdir.sh***. Gunakan notasi [-d NamaDirektori] dan pilih logical && atau || pada level shell.

```
#!/bin/sh
# file: checkdir.sh
# Usage: checkdir.sh DirectoryName
#
if [ $# -ne 1]
then
echo "Error, usage: checkdir.sh DirectoryName"
exit 1
fi
[ … ] && …
```

3. Dengan shell script ***pph.sh***, hitung PPH per tahun dengan ketentuan sebagai berikut:
   - 10 juta pertama PPH 15%
   - 25 juta berikutnya (sisa) PPH 25%
   - Bila masih ada sisa, maka sisa tersebut PPH 35%

   Contoh :
   Gaji 8 juta
   PPH = 15% * 8 juta
   Gaji 12 juta
   PPH =15% * 10 juta + 25% * (12-10) juta
   Gaji 60 juta
   PPH = 15% * 10 juta + 25% * 25 juta + 25% * (60-10-25) juta

   Debugging : untuk melakukan tracing (debug) gunakan opsi –x pada eksekusi shell.

```
$ sh -x pph.sh
+ echo -n 'Berikan gaji dalam ribuan rupiah : '
Berikan gaji dalam ribuan rupiah : + read gaji
20000

+ pkp=10000
```

```
+ '[' 20000 -le 10000 ']'
++ expr 20000 - 10000
+ gaji=10000
+ pph=1500
+ pkp=25000
+ '[' 10000 -le 25000 ']'
+ pkp=10000
++ expr 1500 + 10000 '*' 25 / 100
+ pph=4000
+ echo 'Pajak Penghasilan = 4000'
Pajak Penghasilan = 4000
```

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

```
start
stop
status
restart
reload
```

   Bila buka dari string tersebut, maka berikan pesan error. Sempurnakan program dibawah ini untuk keperluan tersebut

```
#!/bin/sh
# See how we were called
case "$1" in
start)
        echo "Ini adalah start"
        ;;
stop)
        echo "Ini adalah stop"
        ;;
*)
        echo $"Usage:$0 {start|stop|restart|reload|status}"
;;
esac
return
```

5. Buat sebuah fungsi pada script ***confirm.sh*** yang memberikan konfirmasi jawaban **Y**es, **N**o atau **C**ontinue. Jika jawaban Yes, maka beri nilai balik 0, No = 1 dan Continue = 2. Modifikasi kerangka program berikut untuk memenuhi permintaan tersebut.

```
#!/bin/sh
# Confirm whether we really want to run this service
confirm() {
local YES="Y"
local NO="N"
```

```
local CONT="C"

while :
do
  echo -n "(Y)es/(N)o/(C)ontinue? {Y} "
  read answer
  answer=`echo "$answer" | tr '[a-z]' '[A-Z]'`
  if [ "$answer" = "" -0 "$answer" = $YES ]
  then
       return 0
  elif ….
  then
       return 2
  elif ….
  then
       return 1
  fi
done
}
```

Test fungsi diatas dengan program berikut :

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

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

LAPORAN RESMI:

1. Analisa hasil percobaan yang Anda lakukan.
2. Kerjakan latihan diatas dan analisa hasil tampilannya.
3. Berikan kesimpulan dari praktikum ini.