Operasi pada Proses

Minggu ke 4

Zaid Romegar Mair, ST., M.Cs

PROGRAM STUDI TEKNIK INFORMATIKA
Jl. Kolonel Wahid Udin Lk. I Kel. Kayuara, Sekayu 30711
web:www.polsky.ac.id mail: polsky@polsky.ac.id
Tel. / Fax.: +62 714 321099

Operasi Pada Proses

Terdapat dua operasi pada proses, yaitu pembuatan proses (*process creation*) dan penghentian proses (*process deletion*).

Pembuatan Proses

a. Memberi identitas (nama) pada proses yang dibuat;
b. Menyisipkan proses pada list proses atau tabel proses;
c. Menentukan prioritas awal proses;
d. Membuat PCB;
e. Mengalokasikan resource awal bagi proses tersebut

Pembuatan Proses

Ada beberapa kejadian yang menyebabkan pembuatan suatu proses baru, antara lain:

a. Pada lingkungan batch sebagai tambahan atas pemberian job. Setelah menciptakan proses baru, sistem operasi melanjutkan untuk membaca job selanjutnya.
b. Pada ingkungan interaktif, pada saat user baru saja log in;
c. Sebagai tanggapan atas suatu aplikasi (seperti: mencetak file, sistem operasi dapat menciptakan proses yang akan mengelola pencetakan itu);
d. Proses menciptakan proses lain (child).

Pembuatan proses

- Selama eksekusi, suatu proses mungkin akan membuat suatu proses yang baru. Proses tersebut dinamakan parent, sedangkan proses yang dibuat dinamakan child. Proses pembuatan proses anak membentuk pohon proses.
- Pembagian sumber daya :
  - Parent dan child membagi semua sumber daya yang ada
  - Child menggunakan sebagian dari sumber daya yang digunakan parent
  - Parent dan child tidak membagi sumber daya
- Bentuk eksekusi :
  - Parent melanjutkan eksekusi beriringan dengan children.
  - Parent menunggu hingga beberapa atau seluruh children selesai.
- Bentuk ruang alamat :
  - Child adalah duplikat dari proses parent.
  - Child mempunyai program yang diambil dari dirinya.

Pembuatan Proses

- Pada UNIX, parent akan membentuk child dengan menggunakan system call fork. Setelah pemanggilan fork, parent kembali berjalan secara pararel dengan child. Demikian pula, child dapat memanggil fork untuk membentuk child lainnya. Sistem call exec digunakan setelah system call fork mengganti alamat memori proses dengan program baru
- pada MS-DOS, system call akan memanggil binary file tertentu yang ada pada memori dan mengeksekusinya sebagai child. Parent akan running kembali setelah child selesai eksekusi. Dengan demikian parent dan child tidak dapat berjalan secara pararel.

Operasi Pada Proses Cont…

Penghentian Proses

- Suatu proses berhenti jika telah menyelesaikan pernyataan terakhir, dan meminta pada sistem operasi untuk menghapusnya dengan menggunakan system call exit. Proses mengembalikan semua data (output) ke parent proses melalui system call wait. Kemudian proses dihapus dari list atau tabel sistem, dilanjutkan dengan menghapus PCB.

Penghentian Proses

- Parent dapat menghentikan eksekusi proses child dengan menggunakan system call abort. Proses anak dihentikan parent karena beberapa alasan, antara lain :
  - Child mengalokasikan sumber daya melampaui batas
  - Tugas child tidak dibutuhkan lebih lanjut
  - Parent berhenti, karena system operasi tidak mengijinkan child untuk melanjutkan jika parent berhenti dan terminasi dilanjutkan.

Proses yang saling bekerja sama

- Proses-proses yang dieksekusi oleh sistem operasi mungkin berupa prosesproses yang terpisah (independence) atau proses-proses yang saling bekerja sama (cooperate).
- Proses yang terpisah adalah proses yang tidak berakibat atau diakibatkan oleh eksekusi dari proses lain.
- Proses yang saling bekerja sama adalah proses yang dapat berakibat atau diakibatkan oleh eksekusi dari proses lain.
- Contoh :
- P0 menunggu printer P1 menunggu disk drive
- Apabila proses terpisah, meskipun P1 ada dibelakang P0, namun jika disk drive
- nganggur, P1 bisa dieksekusi terlebih dahulu.
- Jika proses tersebut saling bekerjasama maka eksekusi pada suatu proses akan sangat berpengaruh pada proses yang lain, karena mereka saling berbagi data. Contoh :
- P1: ..., ..., ..., ..., P2, ...
- P2: ..., ..., ..., ..., ..., ...

Proses yang saling bekerja sama

- Keuntungan proses yang saling bekerja sama adalah terjadi pembagian informasi, meningkatkan kecepatan komputasi, proses dapat dibagi dalam modul-modul dan lebih memberikan kenyamanan pada programmer.
- Untuk mengillustrasikan proses-proses yang saling bekerjasama ini digunakan producer-consumer problem.
- Producer adalah suatu proses yang menghasilkan informasi yang akan dikonsumsi oleh consumer.
- Sebagai contoh: program untuk mencetak menghasilkan karakter-karakter yang akan dikonsumsi oleh printer driver; compiler menghasilkan kode assembly yang akan dikonsumsi oleh assembler; assembler menghasilkan objek modul yang akan dikonsumsi oleh leader. Kerja producer dan consumer ini harus disinkronisasikan sehingga consumer tidak akan meminta item yang belum diproduksi oleh producer.

Proses yang saling bekerja sama

- Buffer yang berisi item dapat diisi oleh producer dan dikonsumsi oleh consumer.
  - Unbounded-buffer producer consumer problem tidak menggunakan batasan ukuran di buffer. Consumer dapat selalu meminta item baru, dan producer dapat selalu menghasilkan item-item baru.
  - Permasalahan terjadi pada bounded-buffer producer-consumer dimana buffer yang digunakan mempunyai ukuran tertentu. Consumer harus menunggu jika buffer kosong, dan producer harus menunggu jika buffer penuh.
  - Buffer disediakan oleh OS melalui penggunaan IPC (Inter Process Communication) atau secara eksplisit disediakan oleh programmer dengan menggunakan shared memory

Proses yang saling bekerja sama

Penyelesaian dengan shared memory untuk permasalahan bounded buffer

```
#define BUFFER_SIZE 10
Typedef struct {
. . .
} item;
item buffer[BUFFER_SIZE];
int in = 0;
int out = 0;
```

- Variabel in dan out diinisialisasikan dengan nilai nol. Buffer yang digunakan secara bersama-sama diimplementasikan sebagai larik sirkular dengan 2 pointer logika: in dan out.
- Variabel in menunjukkan posisi kosong berikutnya pada buffer; out menunjukkan posisi penuh pertama pada buffer. Buffer kosong jika in == out dan buffer penuh jika in = (in + 1) % BUFFER_SIZE = out.

Proses yang saling bekerja sama

Penyelesaian dengan shared memory untuk permasalahan bounded buffer

- Proses producer :

```
item nextProduced;
while (1) {
while (((in + 1) % BUFFER_SIZE) == out)
; /* do nothing */
buffer[in] = nextProduced;
in = (in + 1) % BUFFER_SIZE;
}
```

- Proses consumer :

```
item nextConsumed;
while (1) {
while (in == out)
   ; /* do nothing */
nextConsumed = buffer[out];
out = (out + 1) % BUFFER_SIZE;
}
```

Komunikasi Antar Proses

- Komunikasi antar proses adalah mekanisme proses-proses untuk berkomunikasi dan melakukan sinkronisasi aksinya. Komunikasi dilakukan dengan sistem pesan, dimana proses berkomunikasi dengan proses lain tanpa menggunakan variabel yang dishare.
- Fasilitas interprocess communication (IPC) terdiri dari dua operasi :
  - send(pesan) dimana ukuran pesan bisa tetap atau berbeda-beda
  - receive(pesan)
- Terdapat dua bentuk komunikasi antar proses yaitu komunikasi langsung (direct communication) dan komunikasi tak langsung (indirect communication).

Komunikasi Langsung

- Bentuk komunikasi langsung adalah proses melakukan komunikasi langsung ke proses lain.
- Pada komunikasi langsung, harus disebutkan nama proses secara eksplisit.
- **send*(P,pesan); mengirim pesan ke proses P.***
- **receive*(Q,pesan); menerima pesan dari proses Q.***
- Properti yang harus terdapat pada saluran komunikasi terdiri dari :
  - Terdapat sambungan yang dapat bekerja secara otomatis antara tiap pasangan proses yang ingin berkomunikasi.
  - Sambungan tersebut menghubungkan tepat satu pasangan proses yang akan berkomunikasi.
  - Antar tiap-tiap pasangan proses terdapat tepat satu saluran.
  - Sambungan tersebut mungkin bersifat *unidirectional, namun biasanya bidirectional*
- *Contoh : producer-consumer problem*

Komunikasi Tak Langsung

- Pada komunikasi tak langsung pengiriman atau penerimaan pesan dilakukan melalui mailbox (port). Mailbox adalah suatu objek yang mana pesan-pesan ditempatkan oleh proses atau dapat dihapus. Tiap-tiap mailbox memiliki identitas unik. Dua buah proses dapat saling berkomunikasi hanya jika mereka saling menggunakan mailbox secara bersama-sama.
- Properti yang harus disediakan pada saluran komunikasi adalah :
  - Sambungan antara 2 proses diberikan jika antara kedua proses tersebut saling menggunakan mailbox secara bersama-sama.
  - Sambungan tersebut dihubungkan dengan beberapa proses.
  - Antar tiap-tiap pasangan proses yang saling berkomunikasi, ada sejumlah sambungan yang berbeda, tiap-tiap link berhubungan dengan satu mailbox.
  - Sambungan tersebut mungkin bersifat unidirectional, namun biasanya bidirectional.
- Operasi yang terdapat pada system mailbox adalah membuat mailbox baru, mengirim dan menerima pesan melalui mailbox dan menghapus mailbox. Primitif yang terdapat pada komunikasi tak langsung adalah :
- send(A,pesan); mengirim pesan ke mailbox A.
- receive(A,pesan); menerima pesan dari mailbox A.

Komunikasi Tak Langsung

- contoh . P1, P2 dan P3 menggunakan mailbox A bersamasama. P1 mengirim pesan sedangkan P2 dan P3 menjalankan operasi **receive. Proses** mana yang mendapatkan pesan.
- Solusi untuk permasalahan ini dapat menggunaikan salah satu cara di bawah ini :
  - Mengijinkan satu sambungan dihubungkan dengan paling banyak dua proses.
  - Mengijinkan hanya satu proses pada satu waktu mengeksekusi operasi **receive.**
  - Mengijinkan sistem untuk memilih penerima tertentu. Proses pengirim diberitahukan proses mana yang menerima.

Buffering

- tiga cara implementasi antrian pesan tersebut yaitu :
  - Zero Capacity: Antrian memiliki panjang maksimum nol, sehingga tidak ada pesan yang menunggu di link. Pada kasus ini, pengirim pesan harus menunggu penerima pesan menerima pesan yang disampaikan sebelum ia mengirim pesan lagi. Kedua proses ini harus berjalan secara sinkron. Sinkronisasi ini sering disebut dengan istilah rendezvous.
  - Bounded Capacity. Antrian memiliki panjang tertentu (n), sehingga ada paling banyak n pesan yang menunggu di link. Jika antrian tidak dalam keadaan penuh, maka jika ada pesan baru dapat menempati antrian yang paling akhir, sehingga pengirim tidak perlu menunggu lagi untuk melanjutkan eksekusi. Jika antrian dalam keadaan penuh, maka pengirim harus menunggu sampai ada tempat kosong.
  - Unbounded Capacity. Antrian memiliki panjang yang tidak tertentu, sehingga ada sejumlah pesan yang dapat menunggu di link. Pengiriman tidak pernah menunda pekerjaan.

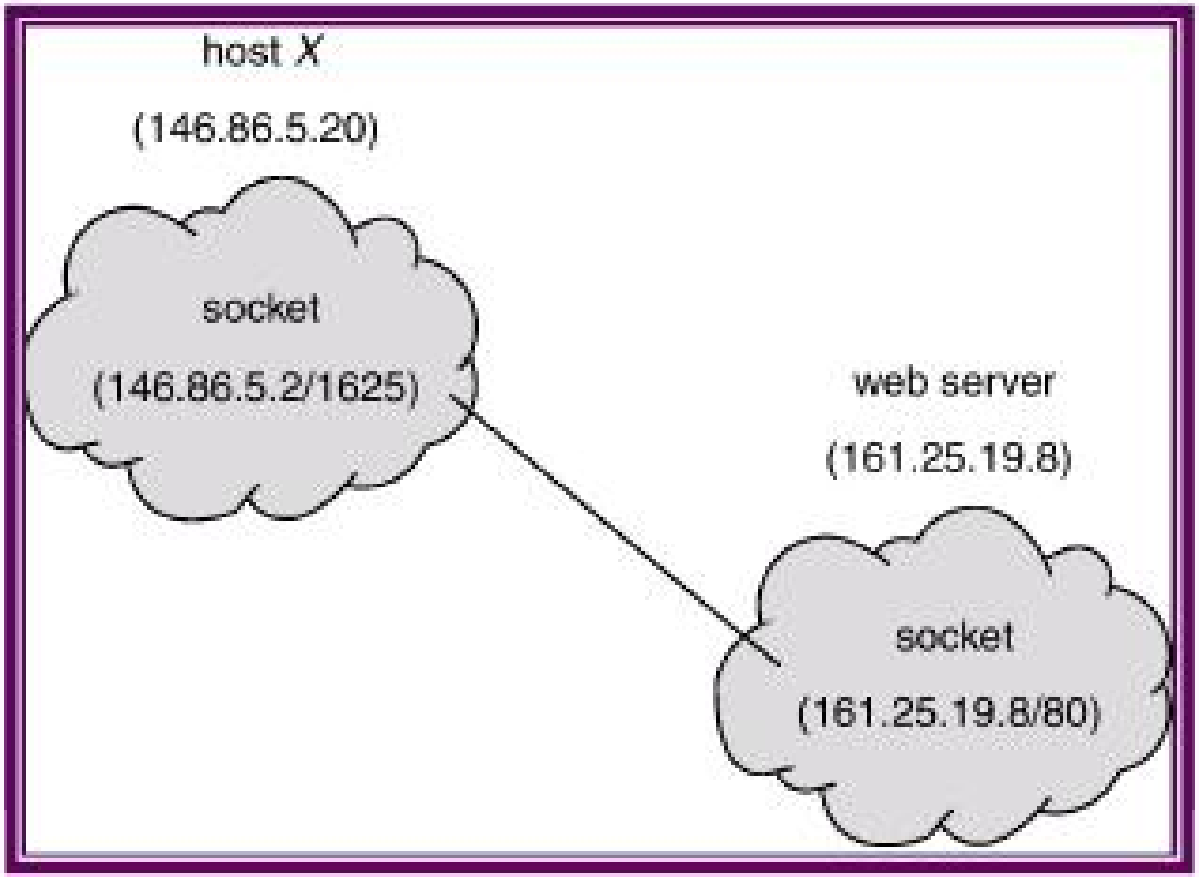
Buffering

- Pada nonzero capacity buffering, proses tidak mengetahui apakah pesan sudah diterima oleh tujuan setelah operasi send selesai
- Jika informasi ini penting untuk komputasi, pengirim harus berkomunikasi secara eksplisit dengan penerima untuk menemukan apakah pesan sudah diterima
- Sebagai contoh misalnya proses p mengirim pesan ke proses Q dan dapat melanjutkan eksekusi hanya setelah pesan diterima

Komunikasi Tak Langsung

- Contoh komunikasi antar proses adalah pada system client server. Komunikasi client server menggunakan berbagai bentuk antara lain socket, remote procedure call (RPC) dan remote method invocation (RMI).
- Socket didefinisikan dengan gabungan antara alamat IP dan port, misalnya socket
- **161.25.19.8:1625 mengacu ke port 1625 pada host 161.25.19.8.**

Komunikasi dengan Socket

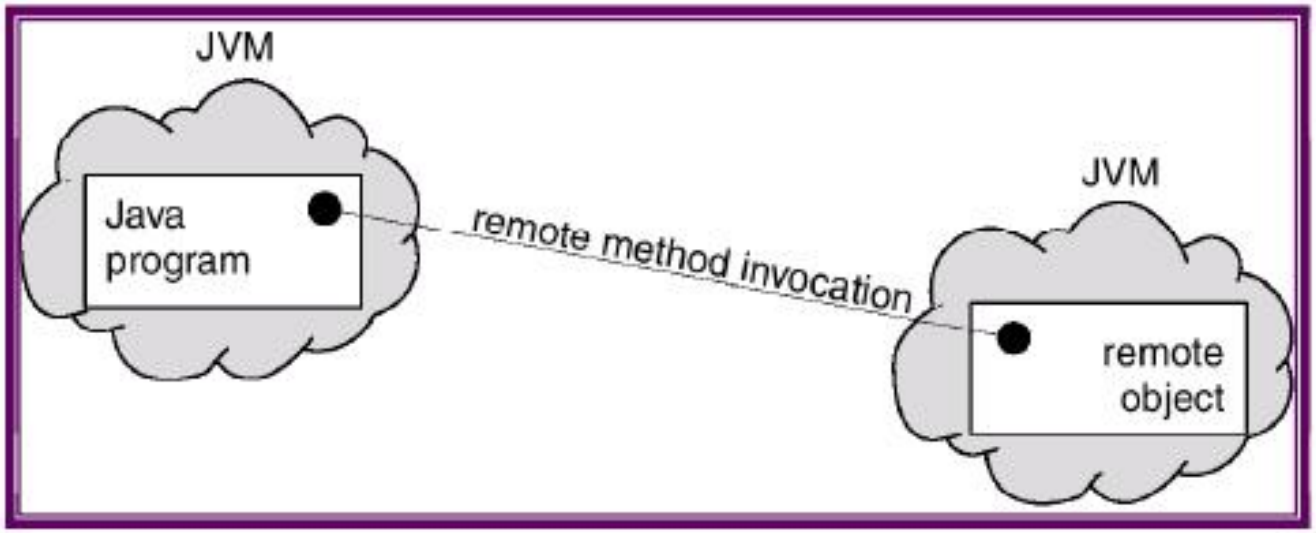


Komunikasi dengan RPC

- Remote procedure call (RPC) merupakan prosedur pemanggilan abstrak antar proses-proses pada system jaringan. Pada RPC terdapat **stub yaitu proxy pada sisi** client untuk prosedur aktual ke server. Stub pada sisi client menghubungi server dan melewatkan parameter, kemudian stub pada sisi server menerima pesan tersebut, menerima parameter dan membentuk prosedur untuk proses server

Komunikasi dengan RMI

- Pada bahasa pemrograman Java terdapat *remote method invocation (RMI) yang* merupakan mekanisme untuk berkomunikasi pada jaringan yang mempunyai bentuk yang sejenis dengan RPC. RMI memungkinkan program Java pada satu mesin mengirim dan menerima method dari obyek secara remote



Praktikum Sistem Operasi
Input dan Output

Minggu ke 4

Zaid Romegar Mair, ST., M.Cs

PROGRAM STUDI TEKNIK INFORMATIKA
Jl. Kolonel Wahid Udin Lk. I Kel. Kayuara, Sekayu 30711
web:www.polsky.ac.id mail: polsky@polsky.ac.id
Tel. / Fax.: +62 714 321099

Proses I/O

- Instruksi (*command*) yang diberikan pada Linux melalui Shell disebut sebagai *eksekusi program* yang selanjutnya disebut *proses.*
- Setiap kali instruksi diberikan, maka Linux kernel akan menciptakan sebuah proses dengan memberikan nomor PID (*Process Identity*).
- Proses dalam Linux selalu membutuhkan Input dan menghasilkan suatu Output.
- Dalam konteks Linux input/output adalah :
  - Keyboard (input)
  - Layar (output)
  - Files
  - Struktur data kernel
  - Peralatan I/O lainnya (misalnya Network)

Pembelokan *(Redirection)*

- Standart input adalah masukkan atau input standard dari suatu perintah atau program. Input standard ini adalah keyboard.
- Standard output adalah keluaran atau output standard dari suatu perintah atau program. Output standard ini adalah monitor atau terminal.
- Standard error adalah keluaran atau output standard jika pada program atau perintah terjadi error. Keluaran ini berupa pesan-pesan kesalahan yang berguna bagi pembuat program atau orang lain yang membutuhkan. Standard error biasanya adalah layar console.

Pembelokan *(Redirection)* Cont…

- Standard input, output dan error, yaitu untuk mengalihkan file descriptor dari 0, 1 dan 2. Linux berkomunikasi dengan file melalui file descriptor yang direpresentasikan melalui angka yang dimulai dari 0, 1, 2 dan seterusnya. Tiga buah file descriptor standar yang lalu diciptakan oleh proses adalah :
- 0 = keyboard (standar input) 1 = layar (standar output) 2 = layar (standar error) Simbol untuk pembelokan adalah :
  - 0 < atau < pengganti standard input
  - 1 > atau > pengganti standard output
  - 2 > atau > pengganti standard error

Pipa *(Pipaline)*

- 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

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 :

$$P1 \mid P2 \mid P3 \ldots\ldots \mid P_{n-1} \mid P_n$$

- 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.

Filter Cont …

- Perintah grep : Digunakan untuk menyaring masukannya dan menampilkan baris-baris yang hanya mengandung pola yang ditentukan. Pola ini disebut *regular expression*.
- Perintah wc : Digunakan untuk menghitung jumlah baris, kata dan karakter dari baris-baris masukan yang diberikan kepadanya. Untuk mengetahui berapa baris gunakan option –l, untuk mengetahui berapa kata, gunakan option –w dan untuk mengetahui berapa karakter, gunakan option –c. Jika salah satu option tidak digunakan, maka tampilannya adalah jumlah baris, jumlah kata dan jumlah karakter.

Filter Cont …

- Perintah sort : Digunakan untuk mengurutkan masukannya berdasarkan urutan nomor ASCII dari karakter.
- Perintah cut : Digunakan untuk mengambil kolom tertentu dari baris-baris masukannya, yang ditentukan pada option –c.
- Perintah uniq : Digunakan untuk menghilangkan baris-baris berurutan yang mengalami duplikasi, biasanya digabungkan dalam pipeline dengan sort.

Latihan

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

Latihan File Descriptor

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

```
$ ps
```

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]
```

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)

Latihan Pembelokan (*redirection*)

Pembelokan standar output

```
$ cat 1> myfile.txt

Ini adalah teks yang saya simpan

Ke file myfile.txt
```

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

```
$ cat 0< myfile.txt

$ cat myfile.txt
```

Pembelokan standar error untuk disimpan di file

```
$ mkdir mydir          (Terdapat pesan error)

$ mkdir mydir 2> myerror.txt

$ cat myerror.txt
```

Latihan Pembelokan (*redirection*) Cont…

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
```

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
```

Notasi >> (append)

```
$ echo "kata pertama" > surat

$ echo "kata kedua" >> surat

$ echo "kata ketiga" >> surat

$ cat surat

$ echo "kata keempat" > surat

$ cat surat
```

Latihan Pembelokan (*redirection*) Cont…

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!

%%%
```

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
```

Latihan Pembelokan (*redirection*) Cont…

Untuk membelokkan standart output ke file, digunakan operator >

```
$ echo hello

$ echo hello > output

$ cat output
```

Untuk menambahkan output ke file digunakan operator >>

```
$ echo bye >> output

$ cat output
```

Untuk membelokkan standart input digunakan operator <

```
$ cat < output
```

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
```

Latihan Pipa (*pipeline*)

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
```

Latihan Filter

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

Zulkifli

Yulizir

Yudi

Ade

[Ctrl-d]
```

Latihan Filter Cont …

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

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