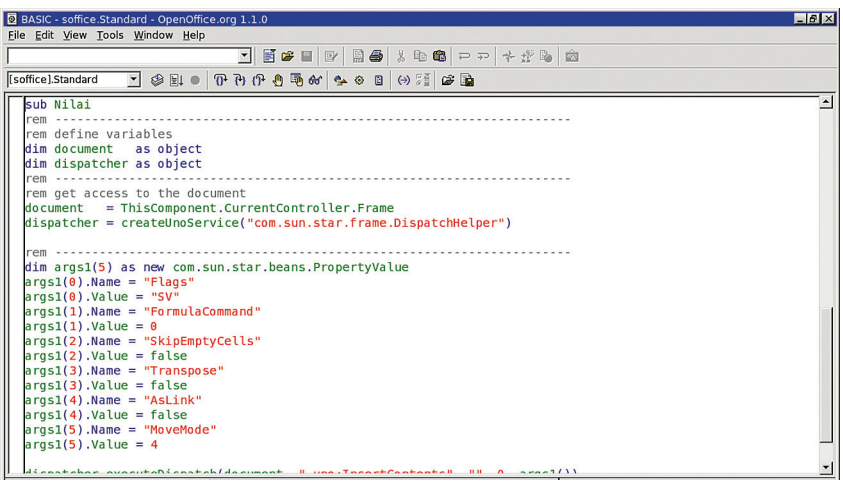
Membuat Macro di OpenOffice.org Calc

Macro sering digunakan untuk memudahkan pekerjaan yang berulang-ulang. Di Microsoft Excel fasilitas ini tersedia dengan bantuan Visual Basic Editor. Bagaimana dengan *spreadsheet* OpenOffice.org? Ternyata sama mudahnya, hanya caranya sedikit berbeda.

Untuk menggunakan macro, Anda tidak harus memahami benar bahasa BASIC. Tetapi jika Anda mengerti dan memahami bahasa BASIC akan lebih baik. Sebagai contoh, penulis agak malas jika harus klik mouse untuk melakukan perintah *Paste Special Value*. Jika memang lebih cepat menggunakan keyboard, mengapa tidak? Cukup dengan menekan kombinasi tombol keyboard pekerjaan tersebut dapat diselesaikan.

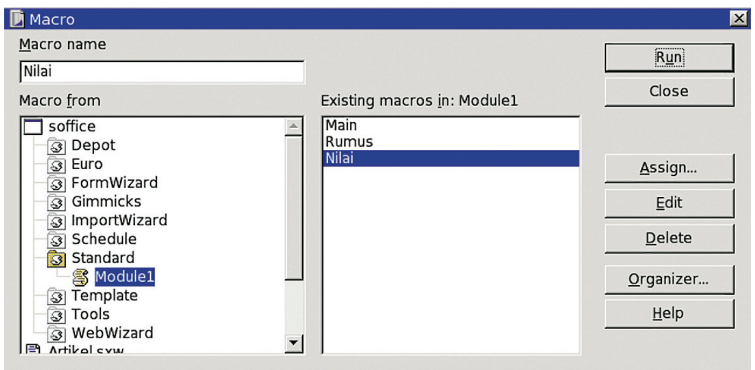
Hal itu memang sederhana, tetapi jika Anda dihadapkan dengan pekerjaan yang banyak mungkin akan sangat membantu. Untuk dapat melakukan perintah tersebut langkah-langkahnya sebagai berikut. Buka jendela editor BASIC (Tools > Macros > Macro), Isikan nama macro di Macro name, misalnya penulis memberikan nama Nilai kemudian klik Edit seperti pada Gambar 1.

Tambahkan baris program berikut untuk perintah Paste Value dari Shortcut keyboard, sehingga tampilan lengkap seperti Gambar 2.

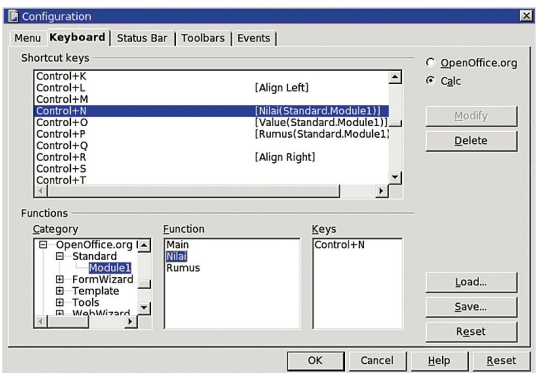


Gambar 2. Menambahkan baris program BASIC di Jendela Editor BASIC.

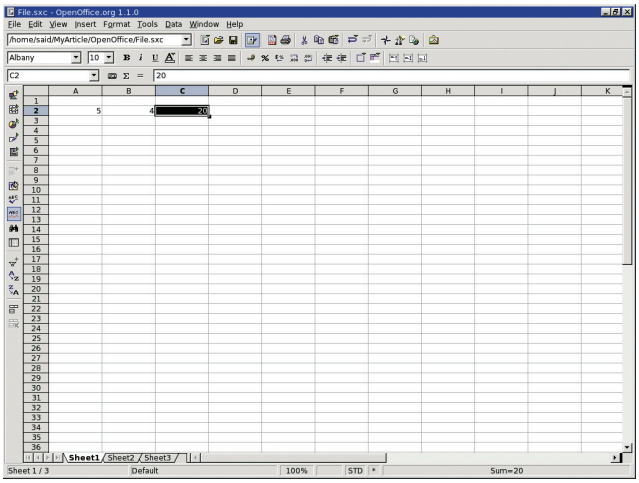
```
rem ----------
rem define variables
dim document   as object
dim dispatcher as object
rem ----------
rem get access to the document
document   = ThisComponent.
CurrentController.Frame
dispatcher = createUnoService("com.
sun.star.frame.DispatchHelper")

rem ----------
```



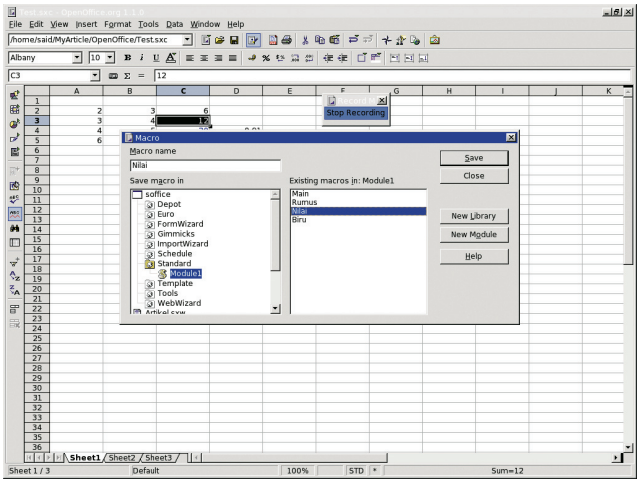
Gambar 1. Membuka jendela Macro.



Gambar 3. Langkah pemilihan shortcut keys.



Gambar 4. Mencoba macro paste value dengan shortcut key.



Gambar 5. Paste value ketika proses recording macro.

```
dim args1(5) as new com.sun.star.
beans.PropertyValue
args1(0).Name = "Flags"
args1(0).Value = "SV"
args1(1).Name = "FormulaCommand"
args1(1).Value = 0
args1(2).Name = "SkipEmptyCells"
args1(2).Value = false
args1(3).Name = "Transpose"
args1(3).Value = false
args1(4).Name = "AsLink"
args1(4).Value = false
args1(5).Name = "MoveMode"
args1(5).Value = 4

dispatcher.executeDispatch(docume
nt, ".uno:InsertContents", "", 0,
args1())
```

Anda sudah selesai membuat macro dengan menggunakan BASIC. Sekarang saatnya Anda membuat Keyboard shortcut-nya. Berikut ini langkah-langkahnya: Tools > Configure, lalu pada Tab Category pilih OpenOffice BASIC Macros > Standard > Module1. Pada Tab Function pilih Nilai, kemudian pada Shortcut keys pilih sesuai dengan keinginan Anda. Penulis memilih Control+N seperti pada Gambar 3.

Mencoba Macro

Untuk mencoba macro yang telah dibuat, penulis mencoba membuat perkalian sederhana di OpenOffice.org seperti pada Gambar 4.

Hasil perkalian tersebut di-*copy* dengan perintah Ctrl+C dan dipaste value dengan perintah Ctrl+N. Mudah, bukan?

Hal ini akan menjadi sulit jika Anda tidak memahami bahasa BASIC. Bagaimana jika Anda tidak memahami bahasa BASIC seperti yang telah penulis janjikan diawal? Jangan khawatir untuk OpenOffice.org versi baru fasilitas *record* macro sudah disediakan sehingga akan memudahkan bagi Anda yang tidak memahami bahasa BASIC.

Berikut ini caranya. Pilih (Tools > Macros > Record Macro) sehingga muncul jendela Stop Recording. Tetapi sebelumnya Anda sudah meng-copy data yang ingin Anda paste value (Hal ini hanya dilakukan untuk inisialisasi pertama), kemudian Anda lakukan paste value dengan melakukan klik mouse seperti biasa (Edit > Paste Special). Beri *check list* hanya pada Strings dan Number untuk paste value seperti pada Gambar 5.

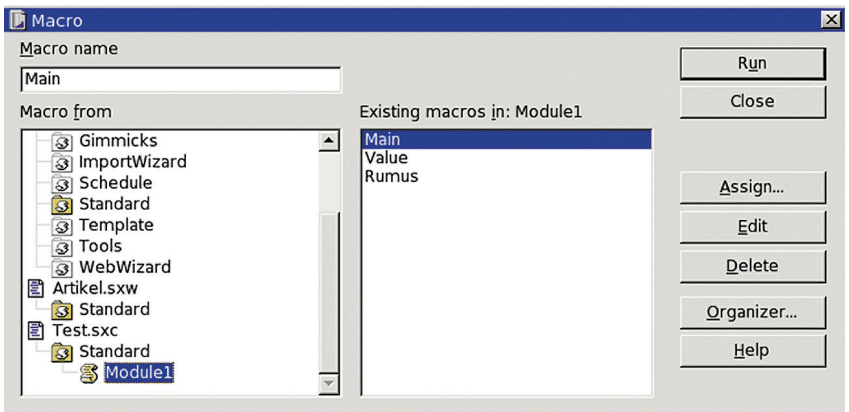
Jika sudah selesai tekan tombol *Stop Recording*. Kemudian beri nama macro yang telah Anda record dan simpan pekerjaan Anda. Setelah itu, berikan *shortcut key* untuk macro yang telah Anda record dengan cara yang sama seperti di atas. Hasilnya sama saja dengan sebelumnya jika Anda ingin melihat program BASIC-nya Anda cukup membuka jendela editor BASIC.

Jika Anda menyimpan source BASIC Anda di soffice Standard, maka macro yang Anda buat dapat berjalan di seluruh file Spreadsheet yang Anda buat. Anda juga dapat membuatnya menjadi lokal saja dengan menaruh source BASIC Anda hanya di file Spreadsheet yang Anda buat, seperti pada Gambar 6.

Perbandingan

Selanjutnya, Anda dapat mengembangkan sendiri macro sesuai dengan kebutuhan. Bahkan untuk OpenOffice.org versi terbaru, tidak hanya bahasa BASIC saja, tetapi juga disediakan bahasa Python dan Java yang memberikan pilihan yang lebih luas bagi Anda. Penulis juga dapat membuat shortcut paste formula. Format penulisan pembuatan macro di Microsoft Excel dan di OpenOffice.org Calc sama mudahnya. Bahkan untuk beberapa hal OpenOffice.org memberikan keleluasaan dalam pembuatannya jika dibandingkan dengan Microsoft Office. Selamat mencoba.

Said Sesiaria [sesiaria@gmail.com]



Gambar 6. Source BASIC di lokal Spredsheet.

Bonding dengan Fedora Core 7

Bonding merupakan suatu cara untuk menggabungkan secara logika beberapa *interface* jaringan menjadi satu. Dengan kata lain, kita akan menggabungkan beberapa *bandwidth* ke dalam satu koneksi. Jadi jika kita menggabungkan dua interface jaringan gigabit, maka sama dengan memiliki satu buah interface jaringan dengan kecepatan 2 gigabit.

Penerapan bonding dapat disesuaikan dengan kebutuhan dalam jaringan Anda karena ada beberapa opsi dalam *bonding*, yaitu:

- Round robin (mode=0 atau balance-rr)
  Mengirim paket secara berurutan mulai dari interface jaringan pertama sampai terakhir.
- Active back-up (mode=1 atau active-backup)
  Hanya satu interface jaringan yang aktif dan interface jaringan yang lain akan aktif bila yang sebelumnya aktif menjadi non aktif.
- Balance XOR (mode=2 atau balance-xor)
  Mengirimkan paket berdasarkan metode alamat MAC asal XOR alamat MAC tujuan.
- Broadcast (mode=3 atau broadcast)
  Mengirimkan paket ke semua interface jaringan
- IEEE Dynamic link (mode=4 atau 802.3ad)
  Merupakan gabungan dari kelompok yang saling berbagi kecepatan dan konfigurasi *duplex* yang sama berdasarkan spesifikasi 802.3ad.
- Adaptive transmit (mode=5 atau balance-tlb)
  Tidak membutuhkan dukungan switch khusus karena jalur keluar disebarkan berdasarkan beban terkini pada tiap interface jaringan. Sedangkan, jalur masuk diterima oleh interface jaringan tersebut. Jika penerimaan interface jaringan tersebut gagal, maka alamat MAC-nya akan diambil alih oleh interface jaringan yang lain.
- Adaptive (mode=6 atau balance-alb)
  Merupakan balance-tlb, namun ditambah dengan *received load balancing* (rlb) untuk jalur Ipv4. Received load balancing didapat melalui negosiasi ARP.

Pada contoh kali ini, akan kita ambil satu opsi bonding sebagai contoh yaitu *round robin* untuk kedua bonding yang akan kita konfigurasikan.

Sedangkan, untuk opsi lain memiliki perlakuan konfigurasi yang sama dan hanya dibedakan dari parameter mode-nya saja.

Kebutuhan software

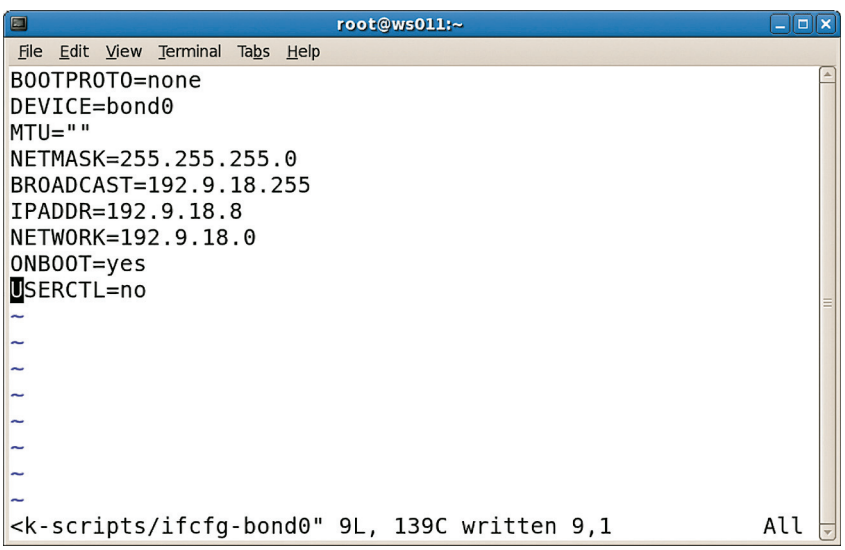
- Linux Fedora Core 7.
- Modul Bonding (standar sudah ada pada distro dan tinggal diaktifkan).

Tahap konfigurasi

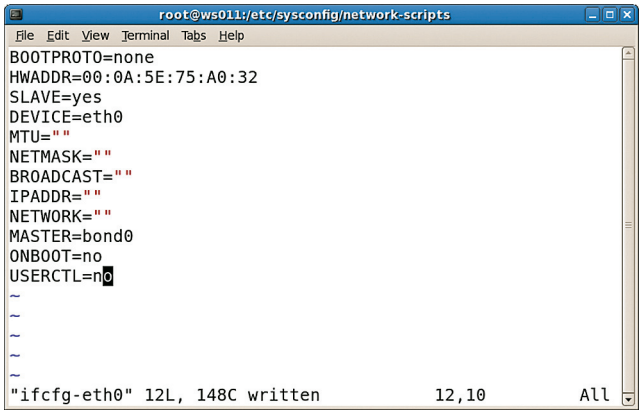
Interface jaringan

Terdapat empat buah interface jaringan yang akan dipasangkan dalam bonding maka pada /etc/sysconfig/network-scripts harus dibuat konfigurasi untuk empat interface jaringan dan dua interface bonding. Berikut ini adalah detail konfigurasinya:

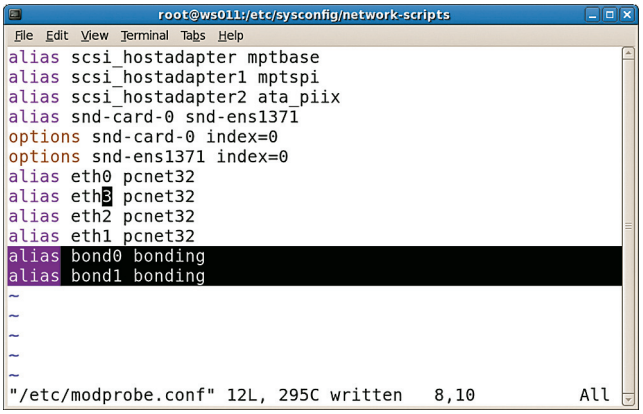
1. Interface bonding
   Jika pada komputer tersebut belum pernah dikonfigurasikan untuk bonding, maka file "ifcfg-bond0" belum



Gambar 1. Isi file /etc/network-scripts/ifcfg-bond0.



Gambar 2. Isi file /etc/network-scripts/ifcfg-eth0.



Gambar 3. Menambah parameter di file /etc/modprobe.conf.

ada. Namun jika sudah ada, maka edit dan konfigurasikan sesuai kebutuhan. Berikut ini contoh konfigurasi untuk ifcfg-bond0 dan ifcfg-bond1.

Pada file ifcfg-bond0 masukkan baris berikut:

```
BOOTPROTO=none
DEVICE=bond0
MTU=""
NETMASK=255.255.255.0
BROADCAST=192.9.18.255
IPADDR=192.9.18.8
NETWORK=192.9.18.0
ONBOOT=yes
USERCTL=no
```

Pada file ifcfg-bond1 masukkan baris berikut:

```
BOOTPROTO=none
DEVICE=bond1
MTU=""
NETMASK=255.255.255.0
BROADCAST=192.9.19.255
IPADDR=192.9.19.254
NETWORK=192.9.19.0
ONBOOT=yes
USERCTL=no
```

2. Interface jaringan (Ethernet)

Jika empat interface jaringan sudah terdeteksi oleh sistem, maka file konfigurasi untuk interface jaringan tidak perlu dibuat, lanjutkan dengan edit dan disesuaikan seperti pada contoh di bawah.

Pada file ifcfg-eth0 masukkan baris berikut:

```
BOOTPROTO=none
HWADDR=00:0A:5E:75:A0:32
SLAVE=yes
DEVICE=eth0
MTU=""
NETMASK=""
BROADCAST=""
IPADDR=""
NETWORK=""
MASTER=bond0
ONBOOT=no
USERCTL=no
```

Pada file ifcfg-eth1 masukkan baris berikut:

```
BOOTPROTO=none
HWADDR=00:04:75:C3:3D:B5
SLAVE=yes
DEVICE=eth1
MTU=""
NETMASK=""
BROADCAST=""
IPADDR=""
NETWORK=""
MASTER=bond0
ONBOOT=no
USERCTL=no
```

Pada file ifcfg-eth2 masukkan baris berikut:

```
BOOTPROTO=none
HWADDR=00:0A:5E:75:9F:C9
SLAVE=yes
DEVICE=eth2
MTU=""
NETMASK=""
BROADCAST=""
IPADDR=""
NETWORK=""
MASTER=bond1
ONBOOT=no
USERCTL=no
```

Pada file ifcfg-eth3 masukkan baris berikut:

```
BOOTPROTO=none
HWADDR=00:0A:5E:75:9F:DF
SLAVE=yes
DEVICE=eth3
MTU=""
NETMASK=""
BROADCAST=""
IPADDR=""
NETWORK=""
MASTER=bond1
ONBOOT=no
USERCTL=no
```

Modul bonding

Sebagaimana kita ketahui bahwa pada kernel dengan dukungan modul bonding, standar maksimum bonding hanya satu. Jadi jika kita akan membuat lebih dari satu bonding, maka modul tersebut harus di-*unload* dan di-*load* kembali dengan tambahan dukungan untuk bonding lebih dari satu (max_bonds=n).

Pada sistem yang akan kita buat kali ini, akan kita gunakan dua bonding jadi max_bonds harus diberikan nilai 2.

- Pada /etc/modprobe.conf tambahkan baris berikut:

```
alias bond0 bonding
alias bond1 bonding
```

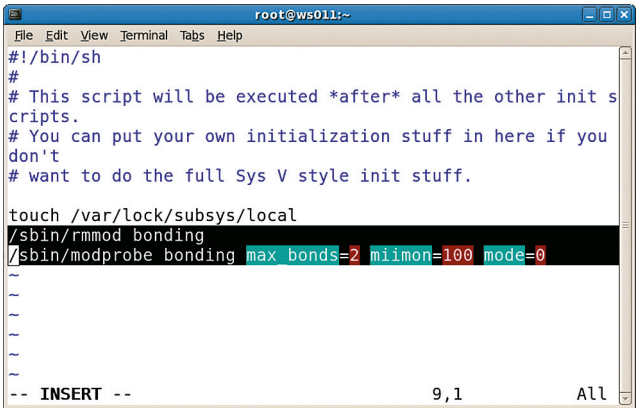
- Pada /etc/rc.d/rc.local tambahkan baris berikut:

```
/sbin/rmmod bonding
/sbin/modprobe bonding max_
bonds=2 miimon=100 mode=0
```

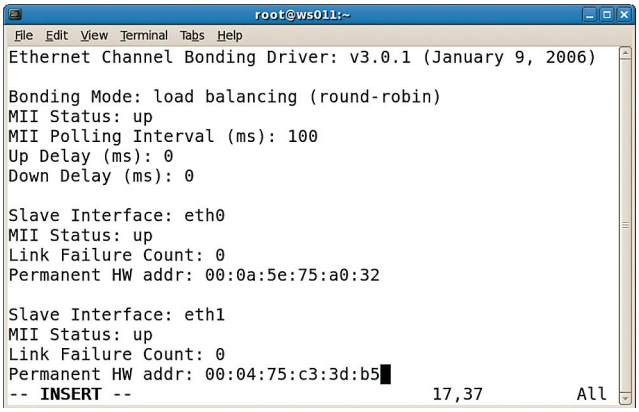
Pengujian

Setelah modul bonding di-*load*, maka pada /proc/net/bonding/ akan terdapat dua buah file yaitu bond0 dan bond1 dengan detail isi sebagai berikut:

```
File /proc/net/bonding/bond0
Ethernet Channel Bonding Driver:
v3.0.1 (January 9, 2006)
```



Gambar 4. Menambah parameter di file /etc/rc.d/rc.local.



Gambar 5. Isi file /proc/net/bonding/bond0.

```
Bonding Mode: load balancing (round-
robin)
MII Status: up
MII Polling Interval (ms): 100
Up Delay (ms): 0
Down Delay (ms): 0

Slave Interface: eth0
MII Status: up
Link Failure Count: 0
Permanent HW addr: 00:0a:5e:75:a0:32

Slave Interface: eth1
MII Status: up
Link Failure Count: 0
Permanent HW addr: 00:04:75:c3:3d:b5


File /proc/net/bonding/bond1
Ethernet Channel Bonding Driver:
v3.0.1 (January 9, 2006)
Bonding Mode: load balancing (round-
robin)
MII Status: up
MII Polling Interval (ms): 100
Up Delay (ms): 0
Down Delay (ms): 0

Slave Interface: eth2
MII Status: up
Link Failure Count: 0
Permanent HW addr: 00:0a:5e:75:9f:c9

Slave Interface: eth3
MII Status: up
Link Failure Count: 0
Permanent HW addr: 00:0a:5e:75:9f:df
```

Aktifasi jaringan bonding

Sebelum kita aktifkan jaringan, sebaiknya pastikan modul bonding sudah di-load dan pengujian berhasil dengan baik. Selanjutnya kita dapat lanjutkan untuk me-*restart* layanan jaringan pada sistem, lakukan perintah berikut ini:

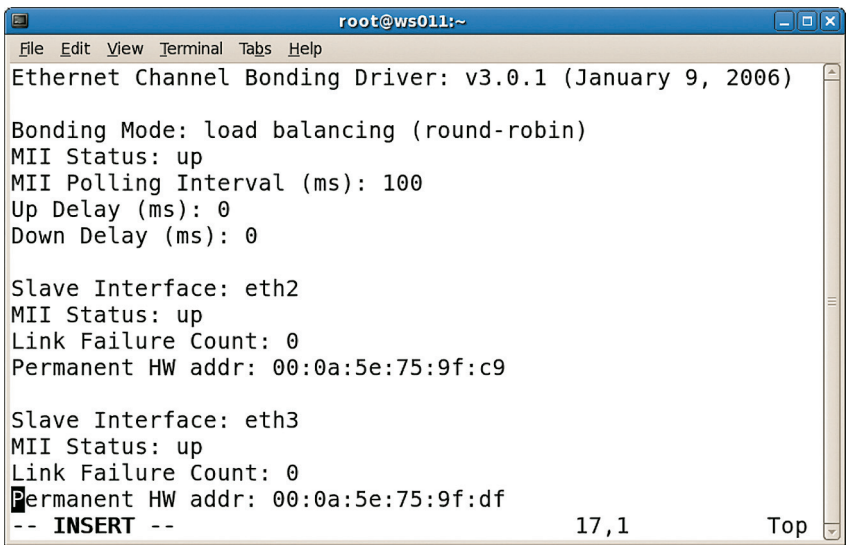
```
# /etc/init.d/network restart
```

Untuk memastikan apakah interface bonding sudah mendapatkan IP address yang diberikan berikan perintah berikut ini untuk melihat statusnya:

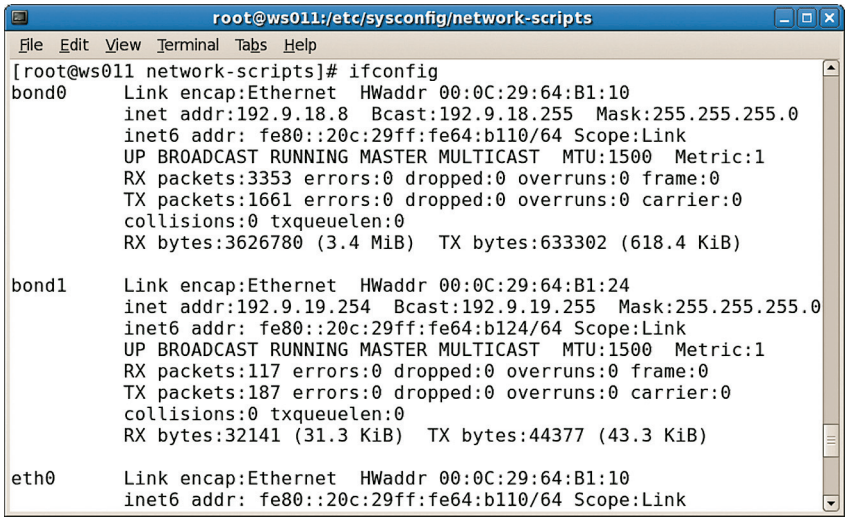
```
# /sbin/ifconfig
```

Lalu akan muncul hasil output seperti di bawah ini/sesuai dengan konfigurasi Anda.

```
bond0     Link encap:Ethernet
HWaddr 00:0A:5E:75:A0:32
          inet addr:192.9.18.8
Bcast:192.9.18.255
Mask:255.255.255.0
          inet6 addr: fe80::200:ff:
fe00:0/64 Scope:Link
          UP BROADCAST RUNNING
MASTER MULTICAST  MTU:1500  Metric:1
          RX packets:818026875
errors:0 dropped:0 overruns:1968
frame:0
          TX packets:656609192
errors:0 dropped:0 overruns:0
carrier:0
          collisions:0 txqueuelen:0
          RX bytes:4022385916 (3.7
GiB)  TX bytes:2922878237 (2.7 GiB)

bond1     Link encap:Ethernet
HWaddr 00:0A:5E:75:9F:C9
          inet addr:192.9.19.254
Bcast:192.9.19.255
Mask:255.255.255.0
          inet6 addr: fe80::200:ff:
fe00:0/64 Scope:Link
          UP BROADCAST RUNNING
MASTER MULTICAST  MTU:1500  Metric:1
          RX packets:666277911
errors:0 dropped:0 overruns:146468
frame:0
```



Gambar 6. Isi file /proc/net/bonding/bond1.



Gambar 7. Tampilan ifconfig setelah selesai konfigurasi network Bonding.

```
          TX packets:752387907
errors:0 dropped:0 overruns:0
carrier:0
          collisions:0 txqueuelen:0
          RX bytes:505913027 (482.4
MiB)  TX bytes:3391926247 (3.1 GiB)

eth0      Link encap:Ethernet
HWaddr 00:0A:5E:75:A0:32
          inet6 addr:
fe80::20a:5eff:fe75:a032/64 Scope:
Link
          UP BROADCAST RUNNING SLAVE
MULTICAST  MTU:1500  Metric:1
          RX packets:408976948
errors:0 dropped:0 overruns:1004
frame:0
          TX packets:328304597
errors:0 dropped:0 overruns:0
carrier:0
          collisions:0
txqueuelen:1000
          RX bytes:1913134455 (1.7
GiB)  TX bytes:3613254894 (3.3 GiB)
          Interrupt:17 Base
address:0x800

eth1      Link encap:Ethernet
HWaddr 00:0A:5E:75:A0:32
          inet6 addr:
fe80::20a:5eff:fe75:a032/64 Scope:
Link
          UP BROADCAST RUNNING SLAVE
MULTICAST  MTU:1500  Metric:1
          RX packets:409049927
errors:0 dropped:0 overruns:964
frame:0
          TX packets:328304595
errors:0 dropped:0 overruns:0
carrier:0
          collisions:0
txqueuelen:1000
          RX bytes:2109251461 (1.9
GiB)  TX bytes:3604590639 (3.3 GiB)
.....
.....
.....
eth3      Link encap:Ethernet
HWaddr 00:0A:5E:75:9F:C9
          inet6 addr:
fe80::20a:5eff:fe75:9fc9/64 Scope:
Link
          UP BROADCAST RUNNING SLAVE
MULTICAST  MTU:1500  Metric:1
          RX packets:333033440
errors:0 dropped:0 overruns:70570
frame:0
          TX packets:376193953
errors:0 dropped:0 overruns:0
carrier:0
          collisions:0
txqueuelen:1000
          RX bytes:169505798 (161.6
MiB)  TX bytes:1691662782 (1.5 GiB)
          Interrupt:20 Base
address:0xac00
```

Selanjutnya Anda dapat aplikasikan konfigurasi tersebut untuk kebutuhan, sesuai dengan opsi yang dibutuhkan. Untuk opsi seperti contoh di atas, penulis telah mencobanya dan telah dilakukan pada File server dan directory server serta PC router. Selamat mencoba!

Sigid [sigidwu@gmail.com]

Membangun Shared Library Sendiri

Ratusan sampai ribuan *shared library* hadir di sistem dan menunjang berbagai aplikasi yang kita gunakan. Ada yang bertugas untuk menangani internal sistem, ada yang bertugas untuk bekerja dengan *hardware* tertentu, menangani terminal, bekerja dengan XML, dan lain sebagainya. Kita pun tentu bisa membangun shared library sendiri, seperti yang kita bahas di tulisan ini.

Shared library merupakan salah satu cara yang digunakan untuk menjadikan program lebih modular, lebih cepat ketika kompilasi dan lebih mudah ketika *update* dilakukan. Suatu program dan program lainnya bisa mempergunakan satu shared library yang sama, sehingga dapat pula menghemat ruang kosong harddisk. Keuntungan lainnya, developer program bisa berkonsentrasi penuh pada program yang dibangun, tanpa harus repot-repot memikirkan fungsionalitas yang telah disediakan oleh shared library tertentu. Bagi pengguna yang terbiasa bekerja dengan Windows, shared library di Linux mirip dengan file DLL di Windows.

Dengan mudah, kita bisa mengenali shared library yang terinstal di sistem kita. Kita bisa membuka direktori /lib, atau /usr/lib, dan akan menemukan banyak sekali file dengan nama yang relatif aneh, yang umumnya diawali dengan lib dan memiliki ekstensi nama file .so.<versi>, dan kemudian terdapat beberapa *symbolic link* ke file tersebut dengan awalan nama yang sama, namun dengan ekstensi yang lebih sederhana. Sebagai contoh:

```
$ ls -al /usr/lib/libncurses.so*
lrwxrwxrwx 1 root root     20 2007-
06-03 20:37 /usr/lib/libncurses.so
-> /lib/libncurses.so.5
lrwxrwxrwx 1 root root     17 2007-
06-03 20:29 /usr/lib/libncurses.so.5
-> libncurses.so.5.5
-rwxr-xr-x 1 root root 267584 2006-
02-08 12:18 /usr/lib/libncurses.
so.5.5
```

Dari contoh tersebut, yang menampilkan pustaka ncurses, dengan nama file libncurses.so.5.5, yang tersimpan di /usr/lib, bisa kita lihat bahwa:

- libncurses.so.5 merupakan symbolic link ke libncurses.so.5.5.
- libncurses.so merupakan symbolic link ke libncurses.so.5.

Sementara, berikut ini adalah tipe file libncurses.so.5.5 menurut program file:

```
$ file /usr/lib/libncurses.so.5.5
/usr/lib/libncurses.so.5.5: ELF 32-
bit LSB shared object, Intel 80386,
version 1 (SYSV), stripped
```

Dari keluaran program file tersebut, file libncurses.so.5.5 adalah shared object (shared library) 32 bit dengan format ELF.

Penjelasan lebih lanjut tentang nama file dan kenapa sistem symbolic link akan kita bahas di bagian detil nama shared library.

Di tulisan ini, kita akan membahas bagaimana kita membangun shared library kita sendiri, termasuk segala sesuatu yang terkait dengan penggunaan shared library.

Untuk memaksimalkan pembahasan, kita tidak akan membahas terlalu detail tentang penggunaan shared library. Pembahasan tentang penggunaan shared library telah kita lakukan pada edisi November 2005.

Semua contoh di tulisan ini dibangun di atas sistem Slackware Linux 11.0, dengan GCC versi 3.4.6, namun seharusnya dapat diterapkan tanpa masalah pada sistem lainnya.

Lebih lanjut dengan nama pustaka

Sebagian besar pustaka umumnya melengkapi dirinya dengan versi mayor, versi minor dan release number. Atau, setidaknya versi mayor dan minor. Mari kita lihat kembali pustaka ncurses yang dibahas sebelumnya, yang tersimpan pada nama file libncurses.so.5.5 (direktori /usr/lib):

- Nama file diakhiri dengan ekstensi .so (*shared object*), diikuti oleh versi mayor, versi minor dan release number (apabila ada). Dalam contoh ini, pustaka ncurses tersebut tidak datang dengan release number.
- Dengan menggunakan penomoran yang jelas, kita bisa mengetahui dengan baik pustaka yang terinstal, terutama kalau pustaka tersebut dirilis cukup sering, dengan perubahan versi yang kecil-kecil.
- Nama file lengkap suatu pustaka, kita sebut sebagai *real name*.
- Ketika kita membangun suatu pustaka, kita perlu menyebutkan secara detail versi pustaka.

Selain real name, kita mengenal pula istilah soname. Soname dari pustaka ncurses sebelumnya adalah libncurses.so.5, yang merupakan symbolic link ke realname libncurs-

es.so.5.5. Berikut ini adalah aturan soname:

- Pola nama umumnya diawali dengan lib, diakhiri dengan .so dan .<versi>.
- Dalam contoh ncurses tersebut, .<versi> adalah .5.
- Ketika suatu program menyatakan membutuhkan suatu pustaka, maka program tersebut umumnya cukup menyatakan membutuhkan soname, bukan realname. Jadi, program membutuhkan libncurses.so.5, bukannya libncurses.so.5.5.

Selain realname dan *soname*, kita mengenal pula linker name. Linker name dari pustaka ncurses sebelumnya adalah libncurses.so, yang merupakan symbolic link ke soname libncurses.so.5. Linker name sendiri adalah symbolic link ke soname atau realname terbaru. Di contoh ini, linker name adalah symbolic ke soname.

Mekanisme penggunaan symlink seperti ini akan memudahkan proses update sistem, termasuk memudahkan pula, apabila program tertentu membutuhkan pustaka versi berbeda.

Melihat kebutuhan pustaka

Di Linux, sangat mudah bagi kita untuk melihat pustaka-pustaka apa saja yang dibutuhkan oleh suatu program. Kita cukup mempergunakan program ldd. Sebagai contoh, kita akan melihat pustaka-pustaka apa saja yang dibutuhkan program program ls (1):

```
$ ldd /bin/ls
  linux-gate.so.1 =>  (0xffffe000)
  librt.so.1 => /lib/tls/librt.so.1
(0xb7ed3000)
  libc.so.6 => /lib/tls/libc.so.6
(0xb7da4000)
  libpthread.so.0 => /lib/tls/
libpthread.so.0 (0xb7d92000)
  lib/ld-linux.so.2 (0xb7eec000)
```

Bisa kita lihat bahwa program ls membutuhkan:

- linux-gate.so.1 (akan dibahas kemudian).
- ld-linux.so.2 (akan dibahas kemudian).
- libc.so.6, yang merupakan pustaka mendasar sistem, yaitu pustaka C (GNU C library).
- libpthread.so.0.
- librt.so.1.

Bisa kita lihat, program ls membutuhkan soname dan bukannya realname. Di sistem penulis, realname dari librt.so.1 adalah librt-2.3.6.so, seperti pada keluaran perintah ls berikut:

```
$ ls -al /lib/tls/librt*
-rwxr-xr-x 1 root root 34763 2006-
09-14 14:54 /lib/tls/librt-2.3.6.so*
lrwxrwxrwx 1 root root    14 2007-
06-03 20:35 /lib/tls/librt.so.1 ->
librt-2.3.6.so*
```

Ketika suatu program membutuhkan pustaka tertentu namun pustaka yang dibutuhkan tidak bisa ditemukan, ldd akan menampilkan not found seperti pada contoh berikut:

```
$ ldd ./test_matematika
  linux-gate.so.1 =>  (0xffffe000)
  libmatematika.so.0 => not found
  libc.so.6 => /lib/tls/libc.so.6
(0xb7dbe000)
  /lib/ld-linux.so.2 (0xb7efe000)
```

Dan, program tersebut pun tidak dapat dijalankan:

```
$ ./test_matematika
./test_matematika: error while
loading shared libraries:
libmatematika.so.0: cannot open
shared object file: No such file or
directory
```

Menarik bukan? Bagi Anda yang ingin melihat detil cara kerja program ldd, bukalah /usr/bin/ldd dengan text editor Anda.

Ld-linux.so.2

Ketika kita menjalankan suatu program di Linux, secara otomatis, program loader juga akan dijalankan. Program loader ini berfungsi untuk menemukan dan menjalankan semua shared library yang dibutuhkan oleh program tersebut.

Program loader di Linux, seperti pada contoh program ls sebelumnya, tersimpan pada /lib/ld-linux.so.2.

Kemanakah loader akan mencari pustaka? Kita akan membahasnya setelah ini.

Lokasi pencarian pustaka

Lokasi pustaka sistem tidak harus selalu di /usr/lib atau di /lib, melainkan di mana saja, walaupun /lib, /usr/lib dan /usr/local/lib merupakan tempat-tempat yang dikhususkan untuk pustaka.

Kita bisa mendefinisikan lokasi pustaka di file /etc/ld.so.conf. Berikut ini adalah contoh /etc/ld.so.conf di sistem penulis:

```
$ cat /etc/ld.so.conf
/usr/local/lib
/usr/X11R6/lib
/usr/i486-slackware-linux/lib
/opt/kde/lib
/usr/lib/qt/lib
```

Tambahkanlah lokasi pustaka Anda di baris baru di file ini, kemudian jalankanlah ldconfig untuk membuat cache pustaka. Cache yang dihasilkan akan disimpan di /etc/ld.so.cache, yang selanjutkan akan di-*load* oleh program loader.

Setiap kali pustaka sistem bertambah, berkurang atau mengalami perubahan, jalankanlah program ldconfig.

Linux-gate.so.1

Ketika kita mempergunakan program ldd untuk melihat kebutuhan pustaka suatu program, ldd akan menampilkan path ke pustaka yang dibutuhkan. Namun, tidak untuk linux-gate.so.1. Bahkan, ketika kita mencari ke file sistem, kita juga tidak akan menemukan file ini.

Lantas, kalau tidak terdapat di file sistem, di mana kita bisa menemukan file ini? Dan mengapa pula kalau file tidak tersedia, ldd tidak menampilkan pesan *not found*?

Linux-gate.so.1 sebenarnya pemetaan ke virtual shared library yang dibuat oleh kernel. Virtual shared library digunakan untuk memilih interface terbaik untuk menjalankan system call tergantung pada CPU.

Dengan demikian, kita tidak perlu mencari keberadaan linux-gate.so.1.

Variabel untuk pencarian pustaka

Seperti disebutkan sebelumnya, lokasi pustaka didefinisikan di /etc/ld.so.conf. Sayangnya, hanya root yang berhak menulis ke file tersebut.

Bagaimana kalau suatu program membutuhkan pustaka tertentu, di mana:

- Lokasinya tidak terdaftar di /etc/ld.so.conf.
- Kita tidak mungkin mengopikan pustaka tersebut ke lokasi-lokasi yang terdaftar di /etc/ld.so.conf.
- Kita tidak memiliki hak root.

Solusinya adalah: gunakan environment variabel LD_LIBRARY_PATH. Aturan penggunaannya sama dengan variabel PATH, yaitu direktori-direktori dideretkan dipisahkan oleh titik dua (:).

Sebagai contoh penggunaan, kita menambahkan direktori aktif (.) dan /mylib ke dalam lokasi pencarian pustaka, dengan tetap

mempertahankan isi variabel LD_LIBRARY_PATH sebelumnya:

```
$ export LD_LIBRARY_PATH=.:/
mylib:$LD_LIBRARY_PATH
```

LD_DEBUG

Environment Variabel yang satu ini sangat berguna ketika kita ingin melihat detail informasi yang berhubungan dengan penggunaan library.

Nilai-nilai yang bisa di-*assign* ke variabel ini:

- files: menampilkan file yang diproses dan pustaka apa yang dibutuhkan.
- bindings: menampilkan informasi tentang *symbol binding*.
- libs: menampilkan *search path* pustaka.
- versions: menampilkan ketergantungan versi tertentu.
- reloc: menampilkan informasi pemrosesan relokasi.
- statistics:menampilkan statistik relokasi.
- unused: menampilkan dynamic shared object yang tidak digunakan.
- all.
- help.

Contoh:

```
$ LD_DEBUG=files /bin/ls
  3127:
  3127: file=librt.so.1 [0];
needed by /bin/ls [0]
  3127: file=librt.so.1 [0];
generating link map
  3127: dynamic: 0xb7eeaeec  base:
0xb7ee4000   size: 0x0000724c
  3127: entry: 0xb7ee5d60  phdr:
0xb7ee4034  phnum:          9
  3127:
  3127:
  3127: file=libc.so.6 [0];  needed
by /bin/ls [0]
  3127: file=libc.so.6 [0];
generating link map
  3127: dynamic: 0xb7edfd5c  base:
0xb7db5000   size: 0x0012ecdc
  3127: entry: 0xb7dc9f10  phdr:
0xb7db5034  phnum:          10
  3127:
  3127:
  3127: file=libpthread.so.0 [0];
needed by /lib/tls/librt.so.1 [0]
  3127: file=libpthread.so.0 [0];
generating link map
  3127: dynamic: 0xb7db1ed0  base:
0xb7da3000   size: 0x000111d8
  3127: entry: 0xb7da7810  phdr:
0xb7da3034  phnum:          9
  3127:
  3127:
  3127: calling init: /lib/tls/
libpthread.so.0
  3127:
  3127:
  3127: calling init: /lib/tls/libc.
so.6
  3127:
  3127:
  3127: calling init: /lib/tls/
librt.so.1
  3127:
  3127:
  3127: initialize program: /bin/ls
  3127:
  3127:
  3127: transferring control: /bin/ls
  3127:
bin  boot  dev  etc  home  lib
media  mnt  opt  proc  root  sbin
sys  tmp  usr  var
  3127:
...
...
  3127: calling fini: /lib/tls/libc.
so.6 [0]
  3127:
```

Mempergunakan bantuan LD_DEBUG=files, kita bisa pula mendapatkan pustaka yang dibutuhkan oleh suatu program:

```
$ LD_DEBUG=files /bin/ls 2> /tmp/
needed
bin  boot  dev  etc  home  lib
media  mnt  opt  proc  root  sbin
sys  tmp  usr  var

$ cat /tmp/needed | grep -i needed
  3162: file=librt.so.1 [0];  needed
by /bin/ls [0]
  3162: file=libc.so.6 [0];  needed
by /bin/ls [0]
  3162: file=libpthread.so.0 [0];
needed by /lib/tls/librt.so.1 [0]
```

Pustaka pertama: libmyhelloworld

Berikut ini, kita akan membangun pustaka pertama kita, yang akan kita berinama libmyhelloworld. Fungsi yang bisa kita pergunakan adalah:

```
void myhelloworld (void);
```

Fungsi tersebut akan mencetak tulisan My Hello World.

Membangun shared library

Untuk membangun pustaka libmyhelloworld, pertama-tama, kita akan menyiapkan dua buah file: myhelloworld.c dan myhelloworld.h.

Berikut ini adalah isi dari myhelloworld.h:

```
void myhelloworld (void);
```

Berikut ini adalah isi dari myhelloworld.c:

```
#include <stdio.h>

void myhelloworld(void)
{
   fprintf (stdout, My Hello World\
n);

}
```

Setelah itu, kita akan melakukan kompilasi:

```
$ gcc -fPIC -c myhelloworld.c
```

Setelah perintah ini dilakukan, kita akan mendapatkan myhelloworld.o. Opsi -fPIC akan mengenable Position Independent Code, yang dibutuhkan oleh sebuah shared library.

```
$ gcc -shared -Wl,-
soname,libmyhelloworld.
so.0 -olibmyhelloworld.so.0.0
myhelloworld.o
```

Setelah perintah ini dilakukan, kita akan mendapatkan libmyhelloworld.so.0.0, yang merupakan pustaka yang ingin kita bangun. Dalam proses kompilasi, kita menyebutkan soname libmyhelloworld.so.0.

Selanjutnya, seperti telah dibahas di awal tulisan, kita membuat soname berupa symlink ke realname libmyhelloworld.so.0.0:

```
$ ln -sf libmyhelloworld.so.0.0
libmyhelloworld.so.0
```

Dan, tidak lupa pula kita membuat linker name, berupa symlink ke soname libmyhelloworld.so.0:

```
$ ln -sf libmyhelloworld.so.0
libmyhelloworld.so
```

Selesai sudah. Pustaka tersebut bisa digunakan seperti halnya shared library lainnya di sistem. Anda bisa mengopikannya ke lokasi yang terdaftar di /etc/ld.so.conf atau mempergunakan variabel LD_LIBRARY_PATH.

Membangun program pengguna

Berikut ini, kita akan membangun program test_myhelloworld, yang akan mempergunakan shared library libmyhelloworld.so.0,

yang kita bangun sebelumnya.

Berikut ini adalah source code test_myhelloworld.c:

```
#include myhelloworld.h

int main(void)
{
   myhelloworld();
   return 0;
}
```

Di dalam source code, kita cukup memanggil fungsi myhelloworld() yang disediakan oleh pustaka libmyhelloworld.so.0.

Selanjutnya, lakukanlah kompilasi dan link dengan perintah berikut:

```
$ gcc -o test_myhelloworld test_
myhelloworld.c -L. -lmyhelloworld
```

Setelah itu, akan terbentuk sebuah program dengan nama test_myhelloworld.

Contoh output:

```
$ LD_LIBRARY_PATH=. ./test_
myhelloworld
My Hello World
```

Melihat kebutuhan pustaka

```
$ LD_LIBRARY_PATH=. ldd test_
myhelloworld
  linux-gate.so.1 =>  (0xffffe000)
  libmyhelloworld.so.0 => ./
libmyhelloworld.so.0 (0xb7f34000)
  libc.so.6 => /lib/tls/libc.so.6
(0xb7df6000)
  /lib/ld-linux.so.2 (0xb7f38000)
```

Pustaka kedua: libmatematika

Pustaka yang kedua ini lebih sedikit kompleks dan melibatkan penggunaan beberapa file source code. Berikut ini adalah fungsi yang disediakan:

```
double kuadrat (double x);
double lebihbesar (double x, double
y);
double lebihkecil (double x, double
y);
```

Fungsi-fungsi tersebut akan bekerja sesuai namanya: menghitung kuadrat, mencari bilangan yang lebih besar dari dua bilangan dan mencari bilangan yang lebih kecil dari dua bilangan.

Membangun shared library

Untuk membangun pustaka, kita akan menyiapkan file:

- matematika.h, header yang akan meng-*include header-header* fungsi yang disediakan. Pengguna yang ingin mempergunakan libmatematika cukup menginclude file ini.
- kuadrat.h dan kuadrat.c
- lebihbesar.h dan lebihbesar.c
- lebihkecil.h dan lebihkecil.c

Berikut ini adalah isi dari kuadrat.h:

```
double kuadrat (double x);
```

Berikut ini adalah isi dari kuadrat.c:

```
double kuadrat (double x)
{
  return x * x;
};
```

Berikut ini adalah isi dari lebihbesar.h:

```
double lebihbesar (double x, double
y);
```

Berikut ini adalah isi dari lebihbesar.c:

```
double lebihbesar (double x, double
y)
{
  if (x > y)
    return x;
  else if (x < y)
    return y;
  else
    return 0;
};
```

Berikut ini adalah isi dari lebihkecil.h:

```
double lebihkecil (double x, double
y);
```

Berikut ini adalah isi dari lebihkecil.c:

```
double lebihkecil (double x, double
y)
{
  if (x < y)
    return x;
  else if (x > y)
    return y;
  else
    return 0;
};
```

Berikut ini adalah isi dari matematika.h:

```
#include kuadrat.h
#include lebihbesar.h
#include lebihkecil.h
```

Lakukanlah kompilasi dengan perintah-perintah berikut:

```
$ gcc -fPIC -c kuadrat.c
$ gcc -fPIC -c lebihbesar.c
$ gcc -fPIC -c lebihkecil.c
```

Setelah kompilasi, akan terbentuk object kuadrat.o, lebihbesar.o dan lebihkecil.o.

```
$ gcc -shared -Wl,-
soname,libmatematika.so.0 -
olibmatematika.so.0.0 kuadrat.o
lebihbesar.o lebihkecil.o
```

Setelah ini, kita akan mendapatkan libmatematika.so.0.0.

Selanjutnya, kita akan membuat symlink untuk soname dan linker name:

```
$ ln -sf libmatematika.so.0.0
libmatematika.so.0
$ ln -sf libmatematika.so.0
libmatematika.so
```

Pustaka libmatematika.so.0.0 pun selesai kita buat dan bisa segera dipergunakan.

Membangun program pengguna

Berikut ini, kita akan membangun program test_matematika, yang akan mempergunakan shared library libmatematika.so.0, yang kita bangun sebelumnya.

Berikut ini adalah source code test_matematika.c:

```
#include <stdio.h>
#include matematika.h

int main(void)
{
  fprintf (stdout, 25 kuadrat = %f\
n, kuadrat(25));
  fprintf (stdout, Bilangan
terbesar antara 10 dan 20 adalah
%f\n, lebihbesar(10,20));
  fprintf (stdout, Bilangan
terkecil antara 10 dan 20 adalah
%f\n, lebihkecil(10,20));

  return 0;
}
```

Selanjutnya, lakukanlah kompilasi dan link dengan perintah berikut:

```
$ gcc -o test_matematika test_
matematika.c -L. -lmatematika
```

Setelah itu, akan terbentuk sebuah program dengan nama test_matematika.

Contoh output:

```
$ LD_LIBRARY_PATH=. ./test_
matematika
25 kuadrat = 625.000000
Bilangan terbesar antara 10 dan 20
adalah  20.000000
Bilangan terkecil antara 10 dan 20
adalah  10.000000
```

Melihat kebutuhan pustaka

```
$ LD_LIBRARY_PATH=. ldd test_
matematika
  linux-gate.so.1 =>  (0xffffe000)
  libmatematika.so.0 => ./
```

```
libmatematika.so.0 (0xb7f2f000)
  libc.so.6 => /lib/tls/libc.so.6
(0xb7df1000)
  /lib/ld-linux.so.2 (0xb7f33000)
```

rpath yang (mungkin) berguna

Selama ini, dalam konteks penggunaan pustaka, kita selalu memandang dari sisi pencarian pustaka oleh program loader. Dengan demikian, kita akan berbicara tentang lokasi pustaka yang ditentukan oleh /etc/ld.so.conf atau variabel LD_LIBRARY_PATH.

Sebenarnya, kita bisa pula mengatur agar lokasi pustaka di-*hardcode* ke dalam program atau pustaka pada saat linking. Sebagai contoh, kita bisa mengatur agar lokasi pencarian di-hardcode ke direktori aktif.

Program contoh yang akan dipergunakan adalah test_matematika. Masih di direktori yang sama, kita akan melakukan kompilasi ulang agar program test_matematika selalu mencari juga ke direktori aktif:

```
$ gcc -Wl,-rpath,. -o test_
matematika test_matematika.c -L. -
lmatematika
```

Sebagai catatan, perbedaan dengan perintah sebelumnya adalah kita menambahkan -Wl,-rpath,. (jangan lupakan titik terakhir).

Setelah perintah ini dilakukan, program test_matematika akan dihasilkan. Berbeda dengan sebelumnya, program ini akan selalu mencari ke direktori aktif sehingga penambahan lokasi pustaka di /etc/ld.so.conf dan variabel LD_LIBRARY_PATH tidak diperlukan.

Bagi pengguna yang ingin mengetahui informasi RPATH yang dikompilasi ke dalam executable, gunakanlah program objdump dengan argumen -p. Contoh:

```
$ objdump -p test_matematika
...
Dynamic Section:
..
  RPATH       .
...
```

Dynamic load shared library

Semua pembahasan sebelumnya memfokuskan pada pembuatan dan penggunaan shared library secara langsung. Dengan demikian, program yang dihasilkan akan 'membutuhkan' shared library tertentu pada saat eksekusi. Apabila shared library tersebut tidak ditemukan, maka program tidak dapat berjalan.

Bagaimana kalau Anda membutuhkan kondisi di mana suatu pustaka adalah opsional. Apabila ada, maka akan di-load. Apabila tidak ada, program tetap dapat berjalan. Hal ini mirip dengan konsep plugin, di mana suatu pustaka di-load hanya kalau dibutuhkan saja.

Untuk memungkinkan hal ini terjadi, kita akan membaca shared library secara dynamic load. Caranya sedikit lebih repot dari contoh penggunaan sebelumnya.

Pustaka contoh yang dipergunakan adalah libmatematika.so.0. Program contoh yang akan membuka pustaka secara dynamic akan disimpan pada dl_test_matematika.c:

```
#include <stdio.h>
#include <dlfcn.h>

int main(void)
{
  void *handle_matematika;

  double (*kuadrat) (double);
  double (*lebihbesar)
(double,double);
  double (*lebihkecil)
(double,double);

  handle_matematika = dlopen(./
libmatematika.so, RTLD_LAZY);

  kuadrat = dlsym(handle_matematika,
kuadrat);
  lebihbesar = dlsym(handle_
matematika, lebihbesar);
  lebihkecil = dlsym(handle_
matematika, lebihkecil);

  fprintf (stdout, 25 kuadrat = %f\
n, (*kuadrat)(25));
  fprintf (stdout, Bilangan
terbesar antara 10 dan 20 adalah
%f\n, (*lebihbesar) (10,20));
  fprintf (stdout, Bilangan
terkecil antara 10 dan 20 adalah
%f\n, (*lebihkecil) (10,20));

  dlclose (handle_matematika);

  return 0;
}
```

Penjelasan:

- Kita tidak mempergunakan matematika.h karena akan melakukan loading secara dynamic. Untuk itu, kita membutuhkan header dlfcn.h.
- Kita menyiapkan beberapa struktur data untuk handle pustaka dan fungsi yang ingin dipergunakan:

```
  void *handle_matematika;
  double (*kuadrat) (double);
  double (*lebihbesar)
(double,double);
  double (*lebihkecil)
(double,double);
```

- Dengan dlopen(), kita bisa membuka pustaka secara dynamic:

```
  handle_matematika = dlopen(./
libmatematika.so, RTLD_LAZY);
```

- Selanjutnya, setelah pustaka dibuka, kita bisa merujuk ke fungsi yang dibutuhkan dengan dlsym():

```
  kuadrat = dlsym(handle_
matematika, kuadrat);
  lebihbesar = dlsym(handle_
matematika, lebihbesar);
  lebihkecil = dlsym(handle_
matematika, lebihkecil);
```

- Dan mempergunakan struktur data yang telah disiapkan sebelumnya, kita bisa mempergunakan fungsi yang telah dirujuk:

```
  fprintf (stdout, 25 kuadrat =
%f\n, (*kuadrat)(25));
```

- Setelah menggunakan, kita menutup dengan dlclose()

```
  dlclose (handle_matematika);
```

Lakukanlah kompilasi dengan perintah:

```
$ gcc -o dl_test_matematika dl_test_
matematika.c -ldl
```

Program ldd tidak menampilkan bahwa kita membutuhkan libmatematika.so.0:

```
$ ldd dl_test_matematika
  linux-gate.so.1 =>  (0xffffe000)
  libdl.so.2 => /lib/tls/libdl.so.2
(0xb7f71000)
  libc.so.6 => /lib/tls/libc.so.6
(0xb7e42000)
  /lib/ld-linux.so.2 (0xb7f86000)
```

Namun, program tetap dapat mempergunakan fungsionalitas libmatematika.so.0:

```
$ ./dl_test_matematika
25 kuadrat = 625.000000
Bilangan terbesar antara 10 dan 20
adalah  20.000000
Bilangan terkecil antara 10 dan 20
adalah  10.000000
```

Seru bukan? Sampai di sini dulu pembahasan kita. Sampai ketemu di kesempatan lainnya.

Noprianto [noprianto@infolinux.co.id]

IKLAN

Bekerja dengan Proses dan Anak Proses

Di Linux, proses bisa memiliki anak proses dan orang tua proses bisa mendelegasikan tugas tertentu kepada anak-anaknya. Dengan demikian, untuk aplikasi besar, kita bisa mendelegasikan sebagian tugas kepada anak. Orang tua bisa mengontrol dan membuat anak lagi apabila diperlukan.

Beberapa edisi belakangan ini, kita telah membahas berbagai *system call* dan contoh pemrograman yang melibatkan proses. Salah satunya adalah bahwa proses bisa saling berkomunikasi dengan proses lainnya dengan mengirimkan signal. Kini, kita membahas yang lebih seru lagi, terutama untuk aplikasi dengan skala lebih besar, di mana proses bisa menciptakan beberapa proses anak, dan kemudian memberikan sebagian pekerjaan kepada anak-anaknya. Cukup mirip dengan kehidupan sehari-hari.

Di tulisan ini, kita akan membahas beberapa teknik menciptakan proses, seperti penggunaan system(), keluarga exec, popen() dan fork(). Beberapa contoh dalam bahasa C dan shell script juga akan kita bahas. Sebelum melanjutkan, apabila perlu, bacalah juga edisi-edisi sebelumnya, tentang penanganan proses di Linux.

Semua contoh di tulisan ini dibangun di atas sistem Slackware 11, kernel 2.6.18, namun seharusnya bisa diterapkan pada sistem lainnya tanpa permasalahan.

Mendapatkan PID orang tua

Kita akan mulai dari anak terlebih dahulu. Proses-proses yang kita buat setelah login ke sistem adalah anak dari proses lain. Dan, jelasnya, merupakan keturunan dari proses init, proses dengan PID 1. Program pstree dapat menampilkan informasi yang lebih jelas secara visual.

```
$ pstree
```

Dalam kondisi di mana kita sebagai anak, perlu mengirimkan pesan tertentu kepada orang tua kita, informasi paling penting yang harus kita ketahui adalah PID milik orang tua. Di pembahasan sebelumnya, kita mengetahui bahwa untuk mendapatkan PID sendiri, kita mempergunakan system call getpid().

Untuk mendapatkan PID milik orang tua, system call yang dipergunakan adalah getppid(). Cara penggunaannya mirip dengan getpid.

Berikut ini adalah s_getppid.c, yang mengilustrasikan contoh penggunaan getppid():

```
#include <stdio.h>
#include <unistd.h>
#include <sys/types.h>

int main(void)
{
  pid_t pid, ppid;

  pid = getpid();

  ppid = getppid();

  fprintf (stdout, PID saya adalah
%d\n, pid);

  fprintf (stdout, PID orang tua
saya adalah %d\n, ppid);

  fprintf (stdout, Tekan CTRL-C
untuk keluar...\n);

  while (1);

  return 0;
};
```

Lakukanlah kompilasi dengan perintah berikut:

```
$ gcc -o s_getppid s_getppid.c
```

Contoh output:

```
$ ./s_getppid
PID saya adalah 2701
PID orang tua saya adalah 2577
Tekan CTRL-C untuk keluar...
```

Biarkanlah program tersebut berjalan, dan periksalah hirarki proses dengan bantuan pstree -p.

Membuat proses dengan system()

Salah satu cara termudah membuat proses adalah dengan mempergunakan fungsi system().

```
#include <stdlib.h>

int system(const char *command);
```

system() akan menjalankan program yang diberikan sebagai argumen dengan menjalankan /bin/sh -c <command>, dan baru akan *return* setelah command selesai dikerjakan.

Berikut ini adalah contoh penggunaan system(), yang disimpan pada s_system.c:

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
```

```
int main(void)
{
  int ret;

  char cmd[20];

  strcpy (cmd, ls -1 /usr/local);

  fprintf (stdout, Menjalankan %s\
n, cmd);

  ret = system (cmd);

  return 0;
};
```

Lakukanlah kompilasi dengan memberikan perintah berikut:

```
$ gcc -o s_system s_system.c
```

Contoh output:

```
$ ./s_system
Menjalankan ls -1 /usr/local
bin
etc
games
include
info
lib
man
sbin
src
```

Membuat proses dengan keluarga fungsi exec

Untuk membuat proses, kita juga bisa mempergunakan fungsi-fungsi yang termasuk dalam keluarga exec:

- execl().
- execlp().
- execle().
- execv().
- execvp().
- execve().

```
  #include <unistd.h>

  extern char **environ;

  int execl(const char *path,
 const char *arg, ...);
  int execlp(const char *file,
 const char *arg, ...);
  int execle(const char *path,
 const char *arg,
    ..., char * const envp[]);
  int execv(const char *path, char
 *const argv[]);
   int execvp(const char *file, char
 *const argv[]);
```

Fungsi-fungsi tersebut akan mengganti process image aktif dengan process image baru. Fungsi-fungsi tersebut akan mengembalikan -1 apabila terjadi kesalahan dan errno akan di-set.

Berikut ini adalah s_execl.c, yang berisikan contoh penggunaan execl():

```
#include <stdio.h>
#include <unistd.h>
#include <errno.h>
#include <string.h>

int main(void)
{

  int ret;

  ret = execl (/bin/ls, ls, -1
, /usr/local, (char *) 0);

  if (ret == -1)
  {
   fprintf (stderr, %s\n, strerror
(errno));
  }

  return 0;
};
```

Cara kompilasi:

```
$ gcc -o s_execl s_execl.c
```

Penjelasan:

- execl() mengharuskan kita menyebutkan path lengkap ke executable
- Argumen diakhiri dengan (char *) 0.

Berikut ini adalah s_execlp.c, yang berisikan contoh penggunaan execlp():

```
#include <stdio.h>
#include <unistd.h>
#include <errno.h>
#include <string.h>

int main(void)
{

  int ret;

  ret = execlp (ls, ls, -1 ,
/usr/local, (char *) 0);

  if (ret == -1)
  {
   fprintf (stderr, %s\n, strerror
(errno));
  }

  return 0;
};
```

Cara kompilasi:

```
$ gcc -o s_execlp s_execlp.c
```

Penjelasan:

- execlp() mempergunakan search path $PATH
- Argumen diakhiri dengan (char *) 0.

Berikut ini adalah s_execle.c, yang berisikan contoh penggunaan execle():

```
#include <stdio.h>
#include <unistd.h>
#include <errno.h>
#include <string.h>

int main(void)
{

  int ret;

  char * nenv[] = {HOME=/tmp,
(char *) 0};

  ret = execle (/bin/ls, ls, -
1, /usr/local, (char *) 0, nenv);

  if (ret == -1)
  {
   fprintf (stderr, %s\n, strerror
(errno));
  }

  return 0;
};
```

Cara kompilasi:

```
$ gcc -o s_execle s_execle.c
```

Penjelasan:

- execle() mengharuskan kita menyebutkan path lengkap ke executable.
- Argumen program dan parameter diakhiri dengan (char *) 0.
- Proses bisa mempergunakan environment yang telah diset di variabel nenv:

```
  char * nenv[] = {HOME=/tmp,
 (char *) 0};
```

Berikut ini adalah s_execv.c, yang berisikan contoh penggunaan execv():

```
#include <stdio.h>
#include <unistd.h>
```

```
#include <errno.h>
#include <string.h>

int main(void)
{

  int ret;

  char * args[] = {ls, -1, /
usr/local, (char *) 0};

  ret = execv (/bin/ls, args);

  if (ret == -1)
  {
   fprintf (stderr, %s\n, strerror
(errno));
  }

  return 0;
};
```

Cara kompilasi:

```
$ gcc -o s_execv s_execv.c
```

Penjelasan:

- execv() mengharuskan kita menyebutkan path lengkap ke executable.
- argumen diberikan dalam bentuk array, diakhiri dengan (char *)0.

Berikut ini adalah s_execvp.c, yang berisikan contoh penggunaan execvp():

```
#include <stdio.h>
#include <unistd.h>
#include <errno.h>
#include <string.h>

int main(void)
{

  int ret;

  char * args[] = {ls, -1, /
usr/local, (char *) 0};

  ret = execvp (ls, args);

  if (ret == -1)
  {
   fprintf (stderr, %s\n, strerror
(errno));
  }

  return 0;
};
```

Cara kompilasi:

```
$ gcc -o s_execvp s_execvp.c
```

Penjelasan:

- execvp() mempergunakan search path $PATH.
- argumen diberikan dalam bentuk array, diakhiri dengan (char *)0.

Berikut ini adalah s_execve.c, yang berisikan contoh penggunaan execve():

```
#include <stdio.h>
#include <unistd.h>
#include <errno.h>
#include <string.h>

int main(void)
{

  int ret;

  char * args[] = {ls, -1, /
usr/local, (char *) 0};
  char * nenv[] = {HOME=/tmp,
(char *) 0};

  ret = execve (/bin/ls, args,
nenv);

  if (ret == -1)
  {
   fprintf (stderr, %s\n, strerror
(errno));
  }

  return 0;
};
```

Cara kompilasi:

```
$ gcc -o s_execve s_execve.c
```

Penjelasan:

- execve() mengharuskan kita menyebutkan path lengkap ke executable
- argumen diberikan dalam bentuk array, diakhiri dengan (char *)0.
- Proses bisa mempergunakan environment yang telah diset di variabel nenv:

```
char * nenv[] = {HOME=/tmp,
(char *) 0};
```

Membuat proses dengan popen()

Mempergunakan popen(), kita bisa membuat proses dengan membuat *pipe*, *forking* (akan dibahas kemudian) dan menjalankan shell.

```
#include <stdio.h>

  FILE *popen(const char *command,
const char *type);

  int pclose(FILE *stream);
```

Karena pipe berlaku satu arah, maka kita harus menentukan proses dibuka untuk baca atau tulis, namun tidak keduanya.

Untuk menutup file, kita mempergunakan pclose().

Berikut ini adalah source code s_popen.c, yang akan memanggil python untuk melakukan perhitungan ekspresi matematika yang dimasukkan oleh user. Output yang dihasilkan oleh program python akan dibuka untuk dibaca dan ditampilkan.

```
#include <stdio.h>
#include <string.h>

int main(void)
{
  char mathexp[20];
  char pycmd[40];
  char result[255];

  FILE *f;

  fprintf (stdout, Masukkan
ekspresi matematika (contoh: 1 +
2): );
  fscanf (stdin, %s, mathexp);

  fprintf (stdout, Anda memasukkan
%s.\nPerhitungan akan dilakukan oleh
Python\n\n, mathexp);

  sprintf (pycmd, python -c 'print
%s', mathexp);

  f = popen (pycmd, r);
  fgets (result, 255, f);
  pclose (f);

  result[strlen(result)-1] = '\0';

  fprintf (stdout, Hasil
perhitungan: %s\n\n\n, result);

  return 0;
};
```

Penjelasan:

- Pembuatan proses python dan pembacaan hasil:

```
sprintf (pycmd, python -c
'print %s', mathexp);

  f = popen (pycmd, r);
```

```
  fgets (result, 255, f);
  pclose (f);

  result[strlen(result)-1] = '\0';

  fprintf (stdout, Hasil
perhitungan: %s\n\n\n, result);
```

Kompilasi:

```
$ gcc -o s_popen s_popen.c
```

Contoh output:

```
$ ./s_popen
Masukkan ekspresi matematika
(contoh: 1 + 2): 12345**50
Anda memasukkan 12345**50.
Perhitungan akan dilakukan oleh
Python
Hasil perhitungan:
37545225181350945210369494233107755
93982215973680203983971888913233738
80424668202771748171426049737415008372
72742104456962110591600569712353910068
68094960606371000010154197834655320757
75709697045385837554931640625
```

Pembuatan satu anak proses dengan fork()

Untuk membuat anak proses, kita bisa mempergunakan system call fork():

```
  #include <sys/types.h>
  #include <unistd.h>

  pid_t fork(void);
```

- Apabila gagal, fork() akan mengembalikan -1 dan errno akan diset.
  - Apabila berhasil, fork() akan mengembalikan nilai dua kali.
  - PID anak proses akan dikembalikan ke orang tua (pembuatan proses).
- 0 akan dikembalikan ke anak.
- fork() bekerja secara asinkron dan kita tidak bisa menebak tugas milik siapa yang akan dikerjakan lebih dahulu. Oleh karena itu, kita tidak boleh melakukan pekerjaan yang saling menunggu, atau saling ketergantungan, yang berpotensi menyebabkan *race condition*.

Berikut ini adalah contoh penggunaan fork(), yang disimpan pada s_fork.c:

```
#include <stdio.h>
#include <unistd.h>
#include <errno.h>
#include <string.h>
#include <sys/types.h>

int main(void)
{
  pid_t child;
  pid_t mypid;

  mypid = getpid();

  int i,j;

  child = fork();

  if ( child == -1)
  {
   fprintf (stderr, %s\n,
strerror(errno));
   return -1;
  }
  else if (child == 0)
  {
   fprintf (stdout, Saya adalah
anak, dengan PID adalah %d\n,
getpid());

   for (i = 1; i <=10; i++)
   {
    fprintf (stdout, \tOutput anak
(%d): %d\n,getpid(), i);
   };

  }
  else
  {
   fprintf (stdout, Saya adalah
orang tua, dengan PID adalah %d\n,
mypid);

   for (j = 1; j<=10; j++)
   {
    fprintf (stdout, \tOutput orang
tua (%d): %d\n,mypid,j);
   };
  };

  return 0;
};
```

Penjelasan:

- Di dalam contoh ini, kita membuat satu anak.
- Pertama-tama, kita memanggil fork().
- Apabila fork() mengembalikan -1, maka program akan diterminasi.
- Apabila fork() mengembalikan 0, yang berarti berada di *execution thread* anak, kita mengerjakan tugas-tugas yang sedianya dikerjakan oleh anak proses:

```
  else if (child == 0)
    {
     fprintf (stdout, Saya
adalah anak, dengan PID adalah
%d\n, getpid());

     for (i = 1; i <=10; i++)
     {
      fprintf (stdout, \tOutput
anak (%d): %d\n,getpid(), i);
     };

  }
```

- Sementara, apabila proses mengembalikan nilai lainnya (positif), maka kita berada pada *execution thread* orang tua dan dapat mengerjakan tugas-tugas orang tua

Lakukanlah kompilasi dengan perintah berikut:

```
$ gcc -o s_fork s_fork.c
```

Contoh output:

```
$ ./s_fork
Saya adalah anak, dengan PID adalah
3012
   Output anak (3012): 1
...
...
   Output anak (3012): 10
Saya adalah orang tua, dengan PID
adalah 3011
   Output orang tua (3011): 1
...
...
   Output orang tua (3011): 10
```

Apabila output pada sistem Anda tampil berantakan (anak dahulu, orang dahulu, atau saling bercampur), hal ini tidaklah masalah karena fork() bekerja asinkron, seperti disebutkan sebelumnya.

Pembuatan dua anak proses dengan fork()

Pembahasan berikut ini adalah pengembangan dari contoh sebelumnya. Di contoh ini, kita akan membuat dua anak. Untuk membuat dua anak, ketika berada di execution thread orang tua, kita mengulangi kembali kode-kode kita seperti ketika kita membuat anak pertama.

Berikut ini adalah source code s_fork3.c:

```
#include <stdio.h>
#include <unistd.h>
```

```
#include <errno.h>
#include <string.h>
#include <sys/types.h>

int main(void)
{
  pid_t child;
  pid_t mypid;

  mypid = getpid();

  int child_1, child_1_i;
  int child_2, child_2_i;

  child = fork();

  if ( child  == -1)
  {
   fprintf (stderr, %s\n,
strerror(errno));
   return -1;
  }
  else if (child == 0)
  {
   for (child_1 = 1; child_1 <=3;
child_1++)
   {
    fprintf (stdout, Saya adalah
anak, dengan PID adalah %d\n,
getpid());
    fprintf (stdout, Tugas ke %d\
n, child_1);

    for (child_1_i = 1; child_1_i
<=3; child_1_i++)
    {
       fprintf (stdout, \tOutput
anak %d: %d\n,getpid(), child_1_i);
    };
   }

  }
  else
  {
   fprintf (stdout, Saya adalah
orang tua, dengan PID adalah %d\n,
mypid);

   if ( (child = fork()) == -1)
   {
    fprintf (stderr, %s\n,
strerror(errno));
    return -1;
   }
   else if (child == 0)
   {
    for (child_2 = 1; child_2 <=2;
child_2++)
    {
       fprintf (stdout, Saya
adalah anak, dengan PID adalah %d\
n, getpid());
       fprintf (stdout, Tugas ke
%d\n, child_2);

       for (child_2_i = 1; child_2_
i <=2; child_2_i++)
       {
                fprintf (stdout,
\tOutput anak %d: %d\n,getpid(),
child_2_i);
       };
    }

   }

  };

  return 0;
};
```

Lakukanlah kompilasi dengan perintah berikut:

```
$ gcc -o s_fork3 s_fork3.c
```

Contoh output:

```
$ ./s_fork3
Saya adalah anak, dengan PID adalah
3037
Tugas ke 1
   Output anak 3037: 1
   Output anak 3037: 2
   Output anak 3037: 3
Saya adalah anak, dengan PID adalah
3037
Tugas ke 2
   Output anak 3037: 1
   Output anak 3037: 2
   Output anak 3037: 3
Saya adalah anak, dengan PID adalah
3037
Tugas ke 3
   Output anak 3037: 1
   Output anak 3037: 2
   Output anak 3037: 3
Saya adalah orang tua, dengan PID
adalah 3036
Saya adalah anak, dengan PID adalah
3038
Tugas ke 1
   Output anak 3038: 1
   Output anak 3038: 2
Saya adalah anak, dengan PID adalah
3038
Tugas ke 2
   Output anak 3038: 1
   Output anak 3038: 2
```

Catatan:

- Apabila tampilan Anda tidak terurut, ingatlah bahwa fork() bekerja asinkron.
- Bisa kita lihat pada contoh output, kita memiliki dua anak, dengan PID yang berbeda-beda (3037 dan 3038), yang mengerjakan tugas sendiri-sendiri.

Menjadi orang tua yang bertanggung jawab

Ketika kita membuat anak proses, kita perlu menunggu anak proses selesai bekerja, untuk kemudian mengetahui *exit* status dari anak tersebut. Dengan menunggu, kita berusaha mencegah terjadinya:

- zombie: anak proses yang selesai sebelum orang tuanya sempat mengetahui. Anak proses masih terdaftar, padahal sebenarnya telah selesai.
- Anak terlantar: orang tua proses selesai sebelum sempat menunggu. Dalam kondisi ini, init akan menjadi orang tua anak proses tersebut.

Untuk menunggu, kita bisa mempergunakan waitpid():

```
   #include <sys/types.h>
   #include <sys/wait.h>

   pid_t wait(int *status);
   pid_t waitpid(pid_t pid, int
*status, int options);
   int waitid(idtype_t idtype, id_t
id, siginfo_t *infop, int options);
```

Di contoh berikut ini, s_fork_wait.c, kita akan membuat anak proses, meminta anak proses melakukan sesuatu, menunggu anak proses selesai bekerja dan mengetahui exit status anak tersebut. Dus, kita menjadi orang tua proses yang bertanggung jawab.

```
#include <stdio.h>
#include <unistd.h>
#include <errno.h>
#include <string.h>
#include <sys/types.h>

int main(void)
{
  pid_t child;
  pid_t mypid;
```

```
  int child_status;

  mypid = getpid();

  int child_1, child_1_i;
  int parent_i;

  child = fork();


  if ( child == -1)
  {
   fprintf (stderr, "%s\n",
strerror(errno));
   return -1;
  }
  else if (child == 0)
  {
   for (child_1 = 1; child_1 <=5;
child_1++)
   {
    fprintf (stdout, "Saya adalah
anak, dengan PID adalah %d\n",
getpid());
    fprintf (stdout, "Tugas ke %d\
n", child_1);

    for (child_1_i = 1; child_1_i
<=5; child_1_i++)
    {
       fprintf (stdout, "\tOutput
anak %d: %d\n",getpid(), child_1_i);
    };
    sleep (2);
   }

  }
  else
  {
   waitpid (child, &child_status,
0);

   fprintf (stdout, Saya adalah
orang tua, dengan PID adalah %d\n,
mypid);
   fprintf (stdout, Anak (pid: %d),
selesai dengan status: %d\n, child,
child_status);

  };

  return 0;
};
```

Lakukanlah kompilasi dengan perintah berikut:

```
$ gcc -o s_fork_wait s_fork_wait.c
```

Contoh output untuk eksekusi normal:

```
$ ./s_fork_wait
...
...
...
Saya adalah anak, dengan PID adalah
3095
Tugas ke 5
   Output anak 3095: 1
...
...
   Output anak 3095: 5
Saya adalah orang tua, dengan PID
adalah 3094
Anak (pid: 3095), selesai dengan
status: 0
```

Contoh output ketika anak proses diterminasi dengan SIGTERM (dengan perintah kill -TERM <PID>):

```
$ ./s_fork_wait
Saya adalah anak, dengan PID adalah
3099
...
...
Saya adalah anak, dengan PID adalah
3099
Tugas ke 2
   Output anak 3099: 1
   Output anak 3099: 2
   Output anak 3099: 3
   Output anak 3099: 4
   Output anak 3099: 5
Saya adalah anak, dengan PID adalah
3099
Tugas ke 3
   Output anak 3099: 1
   Output anak 3099: 2
   Output anak 3099: 3
   Output anak 3099: 4
   Output anak 3099: 5
Saya adalah orang tua, dengan PID
adalah 3098
Anak (pid: 3099), selesai dengan
status: 15
```

Pembuatan anak proses dengan shell script

Bagi kita yang menggemari shell script, anak proses bisa pula dibuat dengan shell script.

Di contoh berikut ini, kita akan menyiapkan dua script, s_parent.sh dan s_child.sh. Yang pertama adalah orang tua proses dan yang terakhir adalah anak proses.

Berikut ini adalah source code s_parent.sh:

```
#!/bin/sh

echo Parent started.

( s_child.sh; ) &

i=1
while [ $i -le  5 ]
do
  echo Output orang tua ($$): $i
  let i=$i+1
  sleep 1
done
```

Berikut ini adalah source code s_child.sh:

```
#!/bin/sh

echo Child started.

i=1
while [ $i -le  5 ]
do
  echo Output anak ($$): $i
  let i=$i+1
  sleep 1
done
```

Penjelasan:

- Tidak ada yang spesial dengan s_child.sh.
- Di s_parent.sh, kita membuat anak proses baru dengan cara:

```
( s_child.sh; ) &
```

Berikanlah hak akses executable pada kedua script tersebut:

```
$ chmod +x s_parent.sh s_child.sh
```

Contoh output:

```
$ ./s_parent.sh
Parent started.
Output orang tua (3125): 1
Child started.
Output anak (3126): 1
Output orang tua (3125): 2
Output anak (3126): 2
Output orang tua (3125): 3
Output anak (3126): 3
Output orang tua (3125): 4
Output anak (3126): 4
Output orang tua (3125): 5
Output anak (3126): 5
```

Sampai di sini dulu pembahasan kita. Selamat mencoba dan mengembangkan!

Noprianto [noprianto@infolinux.co.id]

Menggambar Grafis Vektor Menggunakan Inkscape

CorelDraw dan Macromedia Freehand adalah aplikasi komersial yang cukup banyak digunakan untuk menggambar vektor. Untuk kategori *free software*, sudah tersedia aplikasi sejenis bernama Inkscape yang dapat digunakan secara free. Kali ini akan dijelaskan beberapa teknik penggunaan Inkscape untuk membuat gambar vektor.

Inkscape merupakan aplikasi *free software* untuk membuat gambar berbasis vektor dan memiliki fungsionalitas yang tidak kalah dengan aplikasi sejenis yang komersial. Bahkan saat ini, Inkscape sudah bersifat *multiplatform* sehingga dapat ditemukan pada beberapa sistem operasi. Untuk mengenal lebih jauh mengenai Inkscape dan cara penggunaannya, silakan ikuti penjelasan berikut ini.

Pengertian grafis berbasis vektor dan Grafis berbasis Bitmap

Gambar grafis yang yang bisa diolah ataupun ditampilkan oleh komputer secara umum terbagi atas dua macam, yaitu gambar berbasis vektor dan gambar berbasis bitmap. Gambar yang dihasilkan dengan cara membuat bentuk-bentuk seperti garis, lingkaran, dan kotak akan menghasilkan gambar vektor. Bentuk dasar dari gambar vektor adalah garis lurus dan garis melengkung (kurva). bangun-bangun lain seperti kotak, lingkaran, dan bangun lainnya berasal dari pengembangan garis dan kurva. Contoh perangkat lunak untuk membuat ataupun mengolah gambar berbasis vektor adalah Inkscape dan Corel Draw.

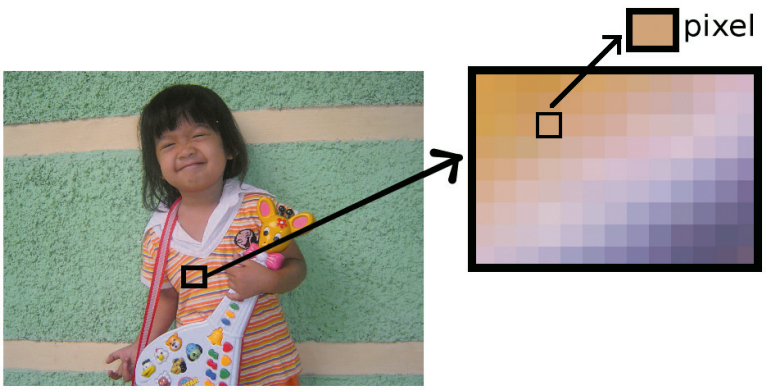
File gambar vektor berisi informasi matematis tentang bangun-bangun yang menyusun sebuah gambar. Tipe file yang lazim digunakan di antaranya adalah SVG (namafile.svg). Sementara gambar yang dihasilkan dari hasil scanning atau hasil transfer dari kamera foto dijital akan menghasilkan grafis berbasis bitmap. Informasi yang terkandung dalam file grafis bitmap berisi informasi titik-titik warna (pixel) yang menyusun sebuah gambar. Sebuah gambar bitmap merupakan kumpulan titik warna yang tersusun rapi membentuk baris dari kiri ke kanan, dan baris-baris tersebut tersusun rapi juga dari atas ke bawah. Dari kumpulan titik itulah terbentuk sebuah gambar bitmap. Contoh perangkat lunak untuk mengolah gambar berbasis bitmap adalah Gimp dan Photoshop. File gambar bitmap berisi informasi warna tiap titik yang menyusun sebuah gambar. Tipe file yang lazim digunakan adalah BMP (namafile.bmp), PNG (namafile.png), JPEG atau JPG (namafile.jpg), dan TIFF (namafile.tif).

Gambar bitmap

Gambar bitmap dibentuk dari kumpulan titik warna yang menyatu membentuk gambar. Setiap titik memiliki warna sendiri-sendiri. Di dalam istilah komputer, titik-titik tersebut disebut sebagai pixel.

Gambar bitmap terbagi dalam beberapa tipe berdasarkan jenis warnanya, yaitu:

1. Black and White: Merupakan tipe gambar berbasis bitmap yang terdiri hanya dari dua warna, yaitu warna hitam dan putih.
2. Grayscale: Merupakan tipe gambar bitmap dengan 256 level gradasi warna dari hitam ke putih.
3. Indexed: Merupakan tipe gambar bitmap yang hanya menampilkan warna-warna tertentu saja. Ada berbagai macam tipe warna Indexed, tergantung pada indeks warna yang digunakan. Indeks warna adalah daftar (atau indeks) warna-warna yang bisa digunakan oleh gambar bertipe Indexed. Jika suatu gambar bitmap diubah dari tipe warna lain ke tipe Indexed, maka warna-warna yang tidak terdapat di dalam indeks warna akan diubah menjadi warna



Gambar 1. Contoh gambar bitmap.

IKLAN

yang terdapat di dalam indeks warna. Web Optimized Palette merupakan salah satu jenis gambar Indexed. Tipe gambar ini hanya mengandung warna-warna yang perlu ditampilkan di website. Warna-warna yang perlu ditampilkan untuk penggunaan di website sudah terangkum dalam indeks warna untuk Web Optimized Palette. Selain Web Optimized Palette, ada banyak jenis tipe bitmap Indexed. Semuanya memiliki indeks warna sendiri-sendiri.

4. RGB atau CMYK: Merupakan tipe gambar yang tiap titik (pixel) nya mempunyai informasi warna dalam bentuk RGB (Red Green Blue atau Merah Hijau Biru) yang lengkap. Tipe gambar ini bisa juga dikatakan sebagai Optimum Palette, karena merupakan jenis gambar yang mengandung semua warna-warna secara lengkap. Bandingkan dengan tipe gambar Indexed yang hanya mengandung warna tertentu saja. Nilai masing-masing Red, Green, dan Blue pada Optimum Palette bisa dari 0 sampai 255 (berarti ada 256 tingkat). Misal untuk pixel berwarna hitam memiliki nilai komponen Red = 0, Green = 0, Blue = 0. Merah murni Red = 255, Green = 0, Blue = 0. Kuning murni Red = 255, Green = 255, Blue = 0. Tipe gambar RGB juga bisa direpresentasikan menggunakan sistem CMYK (Cyan Magenta Yellow blacK) yang memiliki 4 komponen sebagai kombinasi warna. Saat menggunakan RGB warna ditentukan oleh nilai Red, Green, dan Blue sedangkan saat menggunakan CMYK warna ditentukan oleh nilai Cyan, Magenta, Yellow, dan Black.

Sesuai dengan kelengkapan komponen warnanya, konversi warna bisa dilakukan dari yang lebih kompleks ke sederhana. RGB dapat dikonversi ke tipe warna manapun. Misal kita ingin membuat gambar Grayscale dari sebuah gambar bertipe RGB.

Tipe gambar Grayscale bisa dikonversi ke "Black and White", tetapi tidak bisa dikonversi ke RGB. Maksudnya tidak bisa adalah walaupun tipe warna berubah dari Grayscale ke RGB, gambar yang ditampilkan tetap sama seperti semula, yaitu tetap terlihat Grayscale.

Warna Black and White merupakan subset dari warna Grayscale maupun RGB. Warna Grayscale merupakan subset dari warna RGB, tapi menjadi superset dari warna Black and White. Warna RGB merupakan superset dari semua tipe warna. Sementara warna Indexed berkarakteristik sama seperti Grayscale, yaitu menjadi subset dari RGB, tetapi menjadi superset dari warna Black and White dengan syarat warna hitam dan putih termasuk di dalam indeks warnanya.

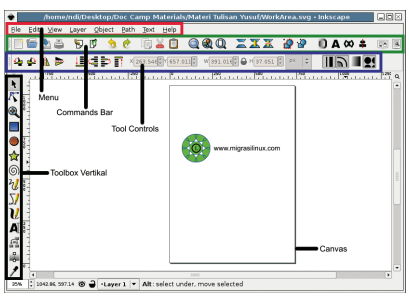
Gambar Vektor

File gambar vektor sepenuhnya disusun berdasarkan rumus-rumus matematika. Misalkan kita membuat sebuah garis, kemudian kita *save* (simpan). Maka yang tersimpan dalam file adalah informasi angka koordinat awal dan angka koordinat akhir, dan warna garis. File gambar vektor tidak menyimpan nilai warna tiap-tiap titik satu per satu. Sebagai hasilnya, file gambar vektor akan berukuran jauh lebih kecil daripada file gambar bitmap karena informasi yang disimpan cukup sedikit saja.

Ketika gambar vektor diperbesar ukurannya (*resize*), gambar akan tetap terlihat halus. Jika dilakukan perbesaran 4 kali, maka secara otomatis software pengolah gambar vektor akan mengalikan nilai-nilai matematis 4 kali nilai semula. Kemudian bentuk-bentuk garis dan kurva penyusun gambar vektor akan digambar ulang dan menghasilkan gambar yang tetap halus dengan ukuran 4 kali ukuran semula.

Pengenalan Inkscape

Fasilitas-fasilitas yang dimiliki Inkscape dalam membuat gambar vektor adalah:



Gambar 2. Working area Inkscape.

- Menu: Berisi berbagai hal menyangkut penggunaan Inkscape.
- Commands Bar: Berisi tombol-tombol short cut ke menu.
- Toolbox (vertikal): Berisi *tool-tool* untuk kebutuhan menggambar.
- Tool Controls: Berisi parameter-parameter untuk menkonfigurasi fungsi tombol Toolbox yang sedang digunakan. Misal saat memilih *Rectangle Tool, Tool Controls* akan menampilkan parameter-parameter yang mengatur Rectangle Tool tersebut.
- Document Canvas: Area tempat menggambar

Teknik dasar

Dasar-dasar pengoperasian Inkscape dalam mengelola file gambar mirip seperti pengoperasian software-software lain. Untuk membuat dokumen kosong baru, dari menu klik File > New atau tekan Ctrl+N. Untuk membuka file dokumen (*.SVG) yang telah disimpan, klik File > Open atau tekan Ctrl+O. Untuk menyimpan gambar yang dibuat ke dalam format SVG, klik File > Save atau tekan Ctrl+S. Untuk menyimpan dengan nama file baru klik File > Save As atau tekan Shift+Ctrl+S .

Membuat bangun segi empat

Untuk menggambar bangun segi empat, pilih Rectangle Tool pada Toolbox vertikal (atau tekan F4). Arahkan mouse ke canvas. Pada posisi tempat kita ingin menggambar segi empat, klik kiri. Sambil tetap ditekan, geser mouse (*drag*) sampai terbentuk segi empat sesuai keinginan kita. Jika bentuk segi empat sudah sesuai dengan keinginan kita, lepaskan mouse. Jika hasil yang didapat masih belum betul-betul sesuai dengan bentuk segi empat yang kita inginkan, tidak perlu khawatir, karena kita masih bisa memodifikasinya sebagaimana yang kita inginkan.

Kita juga bisa menggambar segi empat dengan bantuan tombol Ctrl atau Shift. Pertama-tama tekan Ctrl, lalu klik kiri pada titik awal yang diinginkan. Sambil tetap menekan Ctrl, drag mouse hingga menjadi bentuk yang kita inginkan. Lepaskan Klik kiri baru lepaskan Ctrl. Gambar yang dibuat menggunakan tombol Ctrl selalu memiliki perbandingan panjang dan lebar dalam bilangan bulat (1:1, 1:2, 2:1, 1:3, 3:1, atau seterusnya). Kita buat lagi bangun segi empat baru dengan menekan Shift lalu klik kiri pada titik awal yang diinginkan. Sambil tetap menekan Shift, drag mouse hingga terbentuk segi empat yang kita inginkan. Lepaskan Klik kiri baru lepaskan Shift. Menggambar bentuk menggunakan Shift menyebabkan titik awal akan menjadi titik tengah dari bangun segi empat yang kita buat. Berbeda ketika tidak menggunakan Shift, di mana titik awal merupakan salah satu sudut dari segi empat.

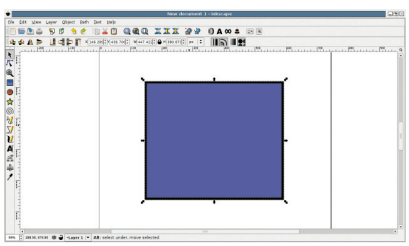
Supaya kita bisa melihat lebih dekat gambar yang telah kita buat, klik Zoom Tool

pada Toolbox vertikal. Klik kiri pada tempat yang diinginkan di canvas untuk melihat lebih dekat (Zoom in). Tekan shift dan klik kiri untuk melihat lebih jauh (Zoom out). Zoom in atau Zoom out tidak akan mengubah gambar yang telah dibuat, hanya memberikan cara pandang terhadap canvas menjadi lebih dekat atau lebih jauh.

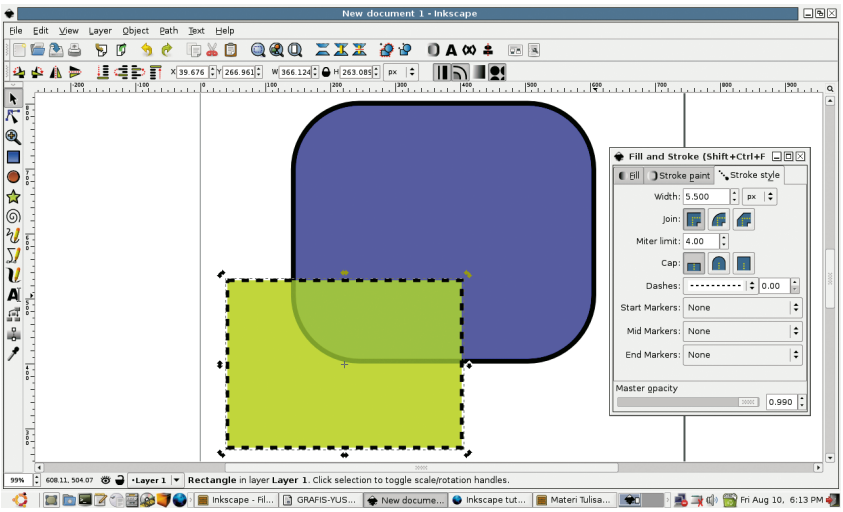
Klik pada tombol Selection Tool pada Toolbox vertikal untuk melakukan modifikasi Moving (menggeser gambar), Scaling (memperbesar/memperkecil gambar), Rotating (memutar gambar). Setelah Selection Tool terpilih, Klik kiri pada gambar yang ingin kita modifikasi. Kita melihat pada gambar terdapat banyak tanda panah yang mengarah ke dalam dan keluar gambar. Saat ini kita berada pada mode Scaling. Klik kiri mouse sekali lagi pada gambar, kita akan melihat tanda panah berubah mengarah ke samping. Saat ini kita berada pada mode Rotating. Untuk berpindah dari mode Scaling ke Rotating dan sebaliknya, cukup dengan melakukan klik kiri pada gambar saja menggunakan Selection Tool.

Pada mode Scaling, arahkan mouse pada salah satu tanda panah hingga tanda panah tersebut berubah warna (menjadi hijau). Klik kiri pada tanda panah tersebut, drag (geser mouse sambil tetap ditekan) ke arah luar untuk memperbesar gambar atau ke arah dalam untuk memperkecil gambar. Setelah mendapat bentuk yang diinginkan lepaskan klik nya. Gambar segi empat yang kita buat telah berubah bentuknya. Lakukan Scaling sambil menekan tombol Ctrl agar perbesaran dilakukan secara proporsional. Pada mode rotating, arahkan cursor mouse ke salah satu tanda panah di sekitar area pilihan (Selection Area). Ketika tanda panah berubah hijau, drag ke samping kiri atau ke samping kanan untuk memutar gambar sesuai keinginan kita. Untuk menggeser gambar, arahkan mouse ke tengah segi empat kemudian klik kiri dan drag gambar ke posisi yang kita inginkan.

Jika kita ingin membuat segi empat yang sudut-sudutnya tidak membentuk sudut



Gambar 3. Melakukan teknik Scaling.



Gambar 4. Metode Fill And Stroke di Inkscape.

menyiku, melainkan membentuk garis lengkung, klik kiri lagi pada Rectangle Tool di Toolbox vertikal. Setelah Rectangle Tool terpilih, pada bagian atas terjadi perubahan pada Tool Controls. Yang sekarang ditampilkan adalah isian-isian angka yang berkaitan dengan Rectangle Tool. Ubah nilai Rx atau Ry dengan menaikan angka di dalamnya. Semakin besar angka yang diisikan, garis lengkung akan semakin membesar. Jika kita ingin mengembalikan ke bentuk sudut menyiku, isi Rx dan Ry harus dibuat 0 atau dengan menekan tombol Not Rounded.

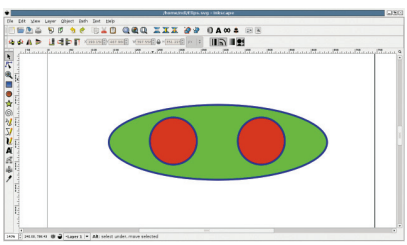
Supaya bisa memahami teknik pengubahan warna di dalam gambar yang kita buat (*fill colour*), warna garis dan ketebalan garis tepi (*stroke*), dan tingkat transparansi, kita perlu membuat segi empat baru yang sebagian bentuknya menimpa gambar yang lain(*overlap*). Lalu pilih segi empat yang baru dibuat (yang berada di atas) menggunakan Selection Tool. Lakukan klik kanan pada segi empat yang baru tersebut, kemudian pilih Fill and Stroke.

Kemudian pilih Fill untuk mengubah warna dan transparansi. Pilihan warna ditentukan oleh nilai R (red/merah), G (green/hijau), dan B (blue/biru). Tingkat transparansi ditentukan oleh nilai A (alpha). Semakin rendah nilai A, warna akan semakin transparan sehingga segi empat lama yang berada di bawah segi empat baru mrnjadi semakin terlihat. Selain pemilihan warna menggunakan metode RGB, terdapat juga pilihan menggunakan metode HSL, CMYK, dan Wheel. Semuanya memiliki manfaat yang sama yaitu memilih warna yang ingin digunakan. Pilih Stroke paint untuk mengubah warna dan transparansi garis tepi. Cara memodifikasinya sama seperti sebelumnya yaitu dengan mengubah-ubah nilai R, G, B, dan A. Pilih Stroke Style untuk mengubah ketebalan dan bentuk garis tepi. Ubah nilai Width untuk mengatur ketebalan garis tepi. Pilih bentuk Dashes untuk mengganti bentuk garis tepi, misal menjadi garis putus-putus.

Untuk menghapus gambar yang tidak dinginkan, gunakan Selection Tool kemudian klik kiri pada gambar yang dimaksud. Setelah gambar tersebut terpilih, tekan Del.

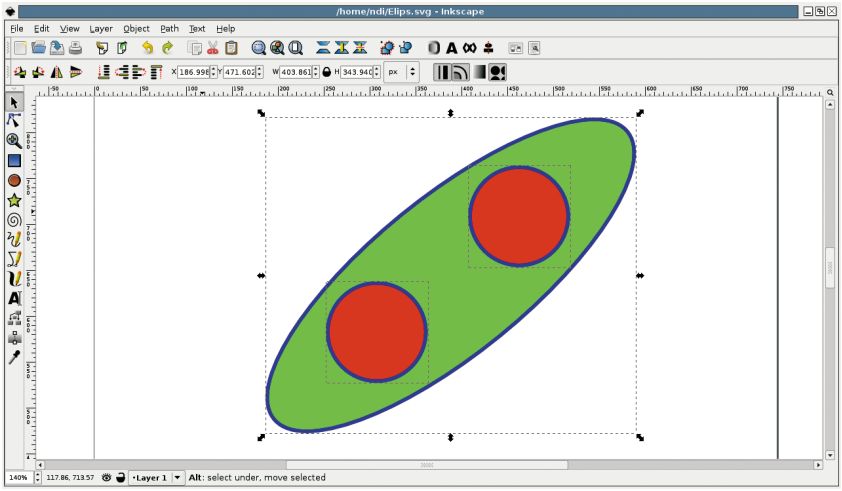
Membuat bangun elips

Klik Ellipse Tool pada Toolbox vertikal untuk menggambar elips atau lingkaran. Klik kiri pada canvas, drag ke posisi akhir, kemudian lepaskan klik. Untuk menggambar lingkaran, tekan Ctrl sebelum mengklik mouse. Sambil tetap menekan Ctrl, klik mouse pada titik awal lalu drag mouse sampai terbentuk lingkaran. Lepaskan klik, baru kemudian lepaskan Ctrl. Jika kita menekan Shift sambil membuat elips atau lingkaran, titik awal klik menjadi titik tengah elips ataupun lingkaran yang dibuat. Gunakan Ellipse Tool untuk menggambar 3 bentuk (1 elips dan 2 lingkaran) seperti berikut:



Gambar 5. Membuat bangun elips.

Sekarang kita akan melakukan Multiple Selection. Klik pada Selection Tool di Toolbox Vertikal. Klik pada salah satu bangun



Gambar 6. Elips Multiple Selection.

lingkaran yang telah kita buat hingga muncul garis putus-putus yang menandakan lingkaran tersebut telah dipilih. Sambil menekan tombol Shift, klik pada lingkaran satu lagi sehingga kedua lingkaran menjadi terpilih. Terlihat bahwa masing masing lingkaran dikelilingi oleh garis putus-putus sendiri-sendiri. Terkahir, lakukan Shift klik seperti tadi pada bangun elips. Kini kita tidak hanya memilih satu bangun saja, tetapi memilih tiga bangun sekaligus.

Untuk memindahkan ketiga bangun tersebut sekaligus, klik lalu drag ke posisi yang diinginkan. Untuk memutar ketiga bangun tersebut, klik satu kali pada salah satu bangun yang terpilih tersebut sehingga tipe Selection berubah dari mode Scaling ke mode Rotating. Sekarang kita bisa memutar (rotate) ketiga bangun tersebut sekaligus dengan cara melakukan klik kemudian drag pada salah satu tanda panah di pojok selection.

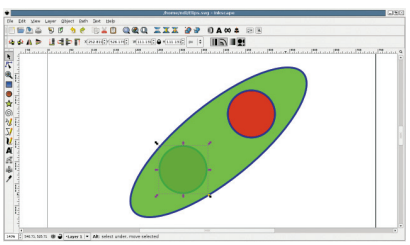
Multiple Selection juga bisa dilakukan dengan cara sambil menekan Shift, lalu klik kemudian drag sehingga seolah-olah membuat sebuah segi empat yang melingkupi bangun-bangun yang ingin dipilih. bangun-bangun yang seluruh bagiannya terlingkupi ketika melakukan drag saja yang akan terpilih.

Untuk membatalkan Multiple Selection, klik mouse di tempat kosong pada canvas di luar bangun-bangun yang terpilih.

Biasanya, setelah melakukan Multiple Selection kita melakukan Grouping. Dengan melakukan Grouping, ketiga bangun yang telah dipilih melalui Multiple Selection tadi dibuat solah-oleh menjadi satu kesatuan. Lakukan Multiple Selection pada ketiga bangun tersebut, kemudian tekan Ctrl G atau dari menu Object > Group. Kini ketiga bangun tersebut seolah-olah menjadi satu bangun saja. Jika kita membatalkan selection dengan mengklik area yang kosong, kemudian melakukan selection lagi pada salah satu bangun tersebut, secara otomatis ketiga bangun tersebut akan terpilih. Hal ini dikarenakan kini ketiga bangun tersebut telah di-*Group* menjadi satu. Untuk memisahkan (*Ungroup*) ketiga bangun tersebut seperti semula, klik pada pada bangun yang telah di-group lalu tekan Shift Ctrl G atau dari menu Object > Ungroup.

Berdasarkan urutan proses menggambar yang sebelumnya dilakukan, elips digambar kali pertama, kemudian lingkaran-lingkaran digambar di atasnya. Kita bisa mengubah urutan posisi tersebut dengan melakukan Raise atau Lower. Klik pada bangun elips kemudian dari menu Object > Raise untuk menaikan posisi urutan elips. Kini posisi bangun elips berada di atas salah satu lingkaran, tetapi elips masih berada di bawah lingkaran yang lain.

Kemudian bagaimana cara kita memilih lingkaran yang berada di bawah elips? misalnya jika kita ingin menggeser lingkaran tersebut. Cara melakukan *Selecting Under* (memilih bangun yang tertutup oleh bangun lain) dengan cara menekan Ctrl Alt dan klik pada lokasi bangun yang ingin dipilih.



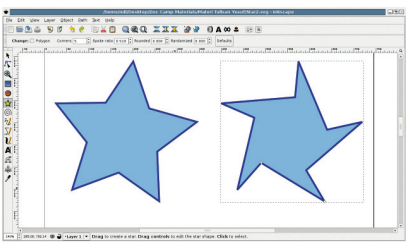
Gambar 7. Elips Select Under.

Jika ada beberapa bangun yang bertumpuk, Ctrl Alt klik pertama akan memilih bangun yang paling atas. Sambil tetap menekan Ctrl Alt, klik berikutnya akan memilih bangun di bawahnya. Ctrl Alt klik berikutnya akan memilih bangun yang di bawahnya lagi, begitu seterusnya hingga bangun yang paling bawah. Setelah bangun yang paling bawah, Ctrl Alt klik berikutnya akan kembali memilih bangun yang paling atas.

Menggambar bangun bintang

Menggambar menggunakan Star Tool bisa untuk membuat berbagai macam bangun yang kompleks dan menarik. Star Tool mempunyai dua tipe dasar: Star dan Polygon. Klik Star Tool pada Toolbox vertikal untuk mengaktifkan Star. Klik pada titik awal yang diinginkan di canvas kemudian drag untuk membuat gambar Star (bintang), lepaskan klik jika ukuran dan arah gambar sudah sesuai keinginan.

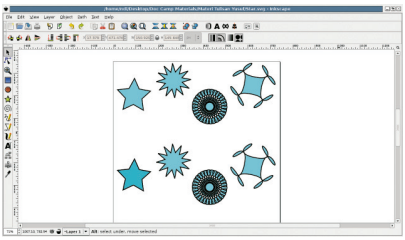
Selama proses drag, kita bisa mengatur besar dan arah bintang. Pada bangun bintang di canvas, terdapat dua titik (*handle*) untuk melakukan modifikasi yaitu Base Radius (posisinya agak di tengah) dan Tip Radius (posisinya di ujung bintang). Klik kemudian drag pada titik Base Radius untuk memodifikasi bagian tengah bintang.



Gambar 8. Star Inner.

Tool Controls (di bagian atas layar, di bawah menu) menampilkan isian-isian berkaitan dengan Star Tool ini. Naikan isian Corners untuk menambah jumlah sudut pada bangun Star. Naikan isian Spoke ratio untuk memperpendek panjang garis sisi-sisi bintang. Naikan isian *Rounded* untuk mengubah sudut-sudut bintang menjadi garis lengkung. Naikan isian Randomized untuk mengubah panjang garis sisi-sisi bintang secara acak menjadi bangun yang tidak beraturan. Mari kita coba buat bangun Star dengan berbagai variasi isian Corners, Spoke ratio, Rounded, dan Randomized.

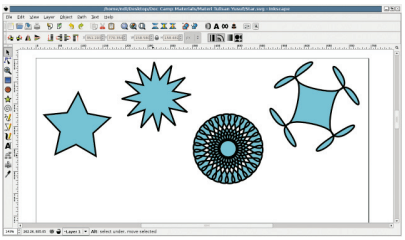
Sedangkan untuk membuat gambar polygon, klik dahulu pilihan polygon pada Tool Controls sebelum mulai menggambar agar kita berpindah dari tipe Star ke tipe Polygon.



Gambar 9. Star Duplicate.

Untuk membantu kita dalam proses menggambar, kita bisa melakukan Duplicate untuk menduplikasi bangun-bangun yang telah kita buat. Duplikasi sangat membantu dalam membuat banyak bangun (terutama yang rumit) supaya sama persis seperti aslinya, tanpa perlu menggambar ulang. Pilih bangun yang ingin di duplikasi, atau lakukan *multiple selection* jika ingin menduplikasi beberapa bangun sekaligus. Tekan Ctrl D atau dari menu Edit > Duplicate. Hasil duplikasi diletakkan tepat di atas aslinya (seolah-olah menjadi satu). Geser hasil duplikasi ke tempat baru yang kita inginkan.

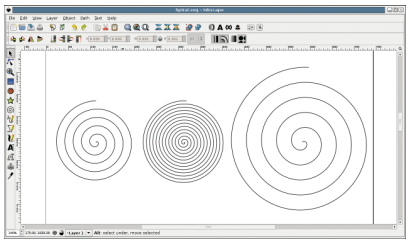
Selain itu ada juga proses Clone yang sama persis seperti proses Duplication. Kelebihan proses Clone adalah jika kita melakukan perubahan pada bangun asli, bangun hasil clone akan mengikutinya. Misalnya kita merubah warna fill colour, melakukan resize, ataupun melakukan rotation. Lakukan Selection pada bangun yang ingin di-clone, lalu tekan Alt D atau dari menu Edit > Clone. Untuk memisahkan keterkaitan bangun asli dengan bangun clone nya, pilih bangun hasil clone menggunakan Selection Tool lalu tekan Shift Alt D.



Gambar 10. Star Many.

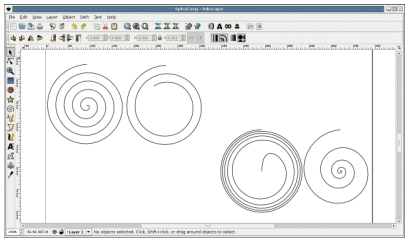
Menggambar Bangun Spiral

Berikutnya adalah menggambar bangun spiral. Klik pada Spiral Tool di Toolbox vertikal. Cara membentuk spiral di canvas seperti membuat bangun-bangun sebelumnya. Setelah bangun spiral jadi, terdapat 2 titik (handle) untuk melakukan modifikasi yaitu Outer Handle (pada ujung terluar spiral) dan Inner Handle (pada titik pusat spiral). Klik kemudian drag pada Outer Handle sambil mengikuti arah putaran spiral untuk menambah jumlah putaran spiral. Untuk menambah jumlah putaran lebih banyak tetapi ukuran spiral tidak bertambah besar, klik mouse pada Outer Handle lalu tekan tombol Alt, kemudian drag sebagaimana menambah jumlah putaran spiral. Hasilnya, jumlah putaran spiral akan bertambah, tetapi ukuran spiral tetap. Sementara jika kita meng-klik pada Outer Handle lalu menekan Shift kemudian men-drag menjauhi/mendekati titik tengah, kita akan memperbesar/memperkecil ukuran spiral.



Gambar 11. Spiral Outer.

Selanjutnya kita bisa melakukan modifikasi menggunakan Inner Handle. Klik pada Inner Handle lalu drag mengikuti putaran spiral, kita akan menghapus putaran spiral mulai dari tengah. Untuk mengembalikannya tekan Shift kemudian klik pada Inner Handle. Inner Handle akan dikembalikan ke posisi semula di tengah. Kita bisa membuat putaran-putaran spiral semakin rapat ke arah luar atau semakin rapat ke arah tengah. Klik pada Inner Handle lalu tekan Alt kemudian drag secara vertika ke atas untuk membuat putaran spiral semakin rapat ke arah luar. Sebaliknya, drag ke bawah untuk membuat putaran spiral semakin rapat ke arah tengah.

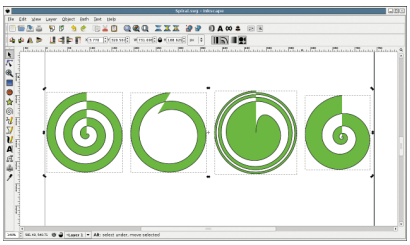


Gambar 12. Spiral Inner.

Bangun-bangun spiral ini *default*-nya dibuat tanpa fill colour. Kita bisa memberikan fill colour dengan lebih dahulu mengklik Select Tool pada Toolbox vertikal, kemudian pilih bangun yang ingin diberi fill colour. Setelah terpilih lalu Klik kanan pada bangun tersebut, kemudian pilih Fill and Stroke. Perhatikan bahwa saat ini yang aktif adalah tombol dengan tanda silang (X) yang artinya *no paint*. Klik pada tombol Flat Colour untuk mengaktifkan fill colour, kemudian pilih warna yang kita inginkan.

Dalam mendesain gambar, sering kali kita ingin mengatur bangun-bangun yang kita buat agar menyusun posisi yang rapi. Proses menyusun bangun-bangun menggunakan teknik *Alignment* dan *Distribution*. Untuk melakukan alignment, lakukan multiple selection terlebih dahulu pada bangun-bangun yang ingin disusun. Setelah itu, tekan Shift Ctrl A atau dari menu Object > Align and Distribute. Terdapat 2 tipe Alignment yaitu Vertical dan Horizontal. Alignment Vertical memiliki beberapa metode penyusunan yang dapat dipilih yaitu rata kiri, rata kanan, dan rata tengah. Sementara Alignment Horizontal memiliki metode penyusunan rata bawah, rata atas, dan rata tengah. Distribution digunakan untuk mengatur jarak antar bangun satu dengan bangun yang lain.

Sebagaimana Alignment, Distribution juga memiliki 2 tipe yaitu Horizontal dan Vertikal. Dengan menggunakan metode Distribution, kita bisa mengatur supaya semua bangun memiliki jarak yang sama satu dengan yang lain. Distribution Horizontal membuat jarak yang sama antar bangun-bangun berdasarkan jarak antar tepi kiri ke tepi kiri, antartitik tengah ke titik tengah, antartepi kanan ke tepi kanan, atau antar-gap.



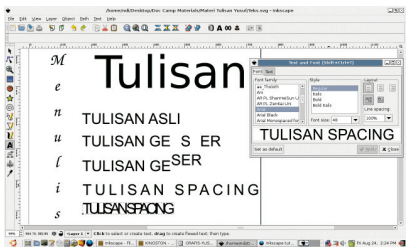
Gambar 13. Spiral Align.

Membuat Teks

Untuk menambahkan tulisan ke dalam gambar yang kita buat, klik pada Text Tool di Toolbox vertikal (atau menekan F8). Klik pada lokasi di canvas yang ingin diberi tulisan. Setelah melihat ada cursor berkedip-kedip, menandakan kita bisa mulai menulis teks yang kita ingin cantumkan. Untuk melakukan perubahan jenis huruf, ukuran huruf, ataupun alignment, klik pada tombol Text and Font Dialog yang ada di Commands Bar (di bawah menu). Atau dari menu Text > Text and Font. Menggunakan fasilitas ini di antaranya kita bisa mengatur alignment tulisan terhadap titik awal cursor apakah sebagai lokasi paling kiri (rata kiri), posisi paling kanan (rata kanan), atau posisi tengah, dari tulisan yang kita buat.

Kita juga bisa mengatur apakah arah tulisan ke samping atau ke bawah. Setiap selesai melakukan perubahan menggunakan Text and Font Dialog, harus diakhiri dengan menekan *Apply* kemudian *Close*.

Teks yang telah kita buat masih dapat kita modifikasi lebih lanjut lagi. Untuk menggeser huruf ke kiri atau ke kanan, letakkan cursor pada huruf yang diinginkan, kemudian tekan Alt Right (Alt dan tombol panah ke kanan) atau Alt Left. Untuk menggeser ke atas dan ke bawah gunakan Alt Up dan Alt Down. Jika kita ingin mengubah jarak spacing antar-huruf, Tekan Alt > atau Alt <. Jarak antarhuruf akan dibuat lebih renggang atau lebih rapat.

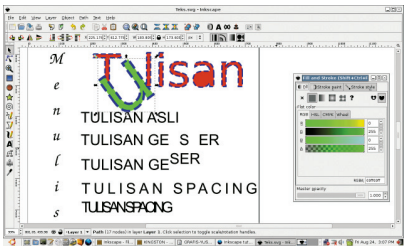


Gambar 14. Membuat Teks.

Yang perlu diperhatikan dalam penulisan teks di Inkscape adalah kita harus yakin jenis huruf yang digunakan tersedia jika file gambar ini disimpan kemudian dibuka di komputer lain. Jika jenis huruf yang digunakan tidak ada di tempat baru, maka saat dibuka nanti, gambar akan mengalami perubahan. Cara mengantisipasi hal ini adalah dengan mengonversi teks yang kita buat menjadi bentuk vektor.

Sebelum melakukan perubahan dari bentuk teks ke bentuk vektor, kita harus memastikan bahwa tulisan yang dibuat sudah sesuai dengan yang kita inginkan dan kita tidak akan melakukan perubahan tebal tipis huruf atau merubah jenis huruf. Pilih tulisan yang ingin dikonversi, lalu dari menu Path > Object to Path. Atau dengan mengklik Shift Ctrl C. Kini teks tersebut sudah berubah menjadi bentuk vektor. Kita bisa mengubah teks sebagaimana merubah bangun kotak atau lingkaran. Sebagai contoh, kini kita bisa merubah fill colour dan garis tepi (stroke). Klik kanan pada tulisan, lalu klik Fill and Stroke. Ubahlah Fill Colour, Stroke Paint, dan Stroke Style.

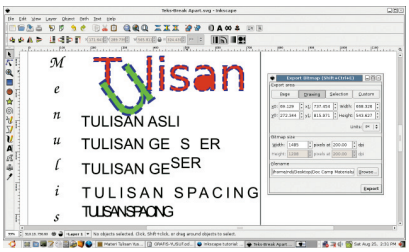
Tulisan hasil konversi ini masih menjadi satu kesatuan. Jika kita melakukan perubahan, akan mempengaruhi semua hurufnya. Kita bisa memisahkannya menjadi masing-masing satu huruf saja menggunakan teknik Break Apart. Klik pada menu Path > Break Apart atau menggunakan short cut Shift Ctrl K. Kini kita dapat melakukan perubahan ke satu huruf saja, karena tiap huruf telah menjadi bangun tersendiri.



Gambar 15. Teks Break Apart.

Sebagaimana sudah diketahui dari awal tulisan ini, bahwa untuk menyimpan gambar yang kita buat, cukup melalui menu File > Save atau File > Save As. Gambar yang kita buat akan disimpan dalam file dengan format SVG.

Selain itu, kita juga bisa melakukan Export Bitmap untuk menyimpan gambar yang kita buat ke dalam format bitmap (Inkscape menggunakan format PNG untuk bitmap). Gambar hasil export tersebut nantinya bisa diedit menggunakan Software Image/Photo Editor seperti The GIMP. Klik menu File > Export Bitmap. Kita dapat memilih Export Area yang ingin kita simpan apakah satu canvas seluruhnya (Page), area canvas yang ada gambarnya (Drawing), area yang bangun-bangunnya sedang dipilih (Selection), atau



Gambar 16. Teks Export.

area yang kita tentukan sendiri (Custom). Untuk semua pilihan tersebut kita diberitahukan titik koordinat awal, koordinat akhir, lebar, dan tingginya. Kita juga bisa mengatur nilai dpi, yaitu jumlah titik (dot atau pixel) tiap incinya. Semakin banyak kita memberikan jumlah titik per inci nya, ukuran gambar akan semakin besar. Pada bagian Filename kita isi dengan nama file berikut path tempat kita ingin menyimpan file bitmap tersebut. Kita akhiri dengan menekan tombol Export.

Yusuf Kurniawan (yusuf132@gmail.com)