

Bahasa Anda di Linux

Noprianto

Ada begitu banyak bahasa pemrograman. Ada begitu banyak GUI toolkit. Mana yang paling memenuhi kebutuhan Anda?

Linux adalah sistem operasi yang mengizinkan setiap penggunanya untuk berpartisipasi dalam pengembangannya. Siapa saja dipersilakan untuk memberikan kontribusinya untuk mewujudkan sistem operasi yang semakin lengkap. Dalam pengembangan aplikasi, layaknya seorang *developer* memerlukan bahasa pemrograman. Angin keterbukaan menyebabkan banyaknya bahasa pemrograman yang dapat digunakan di lingkungan sistem operasi ini. Baik bahasa pemrograman yang sudah bertahan dari zaman dahulu kala yang menawarkan kematangan ataupun bahasa pemrograman yang baru-baru ini lahir dan menjanjikan berbagai kemudahan. Banyaknya bahasa pemrograman ini dari satu sisi menyebabkan sulitnya seseorang untuk memilih mana yang paling cocok untuk memenuhi kebutuhannya.

Untuk itulah *InfoLinux* menghadirkan tulisan ini untuk Anda. Dari sekian luas pilihan yang tersedia, kami mencoba

untuk mengambil yang terbaik untuk dibahas. Pemilihan tersebut pun telah kami sesuaikan dengan kebutuhan dan tren yang telah berlaku cukup lama. Kami tidak bertujuan untuk membandingkan bahasa pemrograman mana yang terbaik, karena masing-masing dari mereka telah memiliki daerah kekuasaan tersendiri. Dari rimba lebat bahasa pemrograman, kami berhasil memilih C/C++, Python, Perl, PHP, java, dan tcl untuk dibahas. Kami juga mencomot GTK+ dan QT dari rimba GUI toolkit. Dan untuk menghargai perbedaan, kami juga menyinggung bahasa pemrograman dan GUI toolkit lainnya. Karena, bukankah perbedaan akan membuat hidup kita semakin berwarna?

Bagaimana kami memilih bahasa-bahasa tersebut? Pertama-tama, kami mencari sesuatu yang sejati dan matang. Untuk itu, kami berhasil menangkap C/C++. Kemudian, kami memasuki daerah bahasa pemrograman yang menawarkan fitur dan kemudahan. Dari sana, kami

berhasil mendapatkan Python, Perl, dan Tcl. Untuk Anda yang akan membuat aplikasi mulai dari level *enterprise* sampai arsitektur mikro, kami langsung jatuh cinta kepada Java. Tidak lupa, kami juga mendapatkan PHP untuk pemrograman web.

Adapun perbandingan kami lakukan pada aspek-aspek yang menurut kami paling relevan. Untuk beberapa hal yang mengganggu lisensi dan aturan perusahaan pembuatnya, kami merujuk ke dokumentasi resmi dari lembaga tersebut. Tidak ada satu pun niat dari kami untuk menjelekkan satu bahasa sementara mengagungkan bahasa lainnya. Beberapa situs yang kami sertakan di tulisan ini juga ditulis tanpa suatu urutan tertentu. Kami sungguh mohon maaf apabila terdapat bagian dari tulisan ini yang menyinggung perasaan Anda.

Sebagai hasilnya, kami ucapkan selamat membaca untuk Anda, pembaca setia majalah *InfoLinux*. 🐧

KATEGORI BAHASA PEMROGRAMAN

■ C /C++



Kalau ada bahasa yang bisa dianggap sebagai orang tua dari bahasa-bahasa lainnya, maka itulah bahasa C.

Bahasa yang ditulis oleh **Dennis Ritchie** dan **Brian Kernighan** ini, pada awalnya bertujuan untuk meningkatkan portabilitas sambil tetap menjaga kemampuannya. Standar bahasa C pada tahun 70-an tersebut telah mengalami berbagai pergeseran sampai saat ini, sesuai kebutuhan zaman. Akan tetapi, karena bahasa C telah memiliki standardisasi, maka terkadang pergeseran tersebut pada umumnya hanya berupa penambahan fitur.

Standar asli ANSI C (X3.159-1989) diratifikasi pada tahun 1989 dan dipublikasikan pada tahun 1990. Standar ini kemudian diratifikasi sebagai standar ISO pada tahun yang sama (ISO/IEC 9899:1990). Secara umum, kedua standar tersebut tidak memiliki banyak perubahan. Standar-sandar tersebut lebih dikenal sebagai C89 ataupun C90. Sayangnya standar ISO ini memiliki kesalahan, yang akhirnya diperbaiki pada tahun 1994 dan 1996.

Karena spesifikasinya yang terbuka, maka terdapat begitu banyak kompiler bahasa C yang terdapat di berbagai platform. Di dunia Linux, Anda bisa menggunakan GCC (*GNU Compiler Collection*) sebagai *compiler* untuk bahasa C. Compiler suite yang satu ini dilisensikan di bawah GPL dan dapat di-download di <http://gcc.gnu.org>. Saat ini, GCC dikontrol oleh GCC Steering Committee yang memiliki anggota dari berbagai kalangan. Hal ini juga menepis kabar burung yang mengatakan bahwa kontrol GCC dipegang oleh RedHat, yang menyumbangkan *hardware*, koneksi jaringan, kode dan waktu pengembangan. GCC sendiri, sebagai suatu koleksi compiler memiliki dukungan untuk C, C++, Objective C,

Fortran, dan Java. Dukungan untuk beberapa bahasa lainnya akan ditingkatkan di masa mendatang. Tulisan ini hanya akan membahas bahasa pemrograman C (dan C++) dari perspektif GCC dan penggunaannya di Linux.

Apabila Python, Perl, PHP, dan Tcl adalah bahasa *scripting* dan berjalan sesuai prinsip kerja interpreter, maka C adalah bahasa pemrograman “sejati” dan berjalan sesuai prinsip kerja compiler. Dengan demikian, setiap program yang Anda tulis hanya dapat digunakan setelah mengalami proses kompilasi dan *linking* yang sukses.

Topik seputar daerah kekuasaan bahasa yang satu ini sangatlah menarik untuk dibahas. Kalau dibandingkan dengan bahasa-bahasa *scripting* lainnya, maka bahasa C adalah penguasa yang sesungguhnya. Bagaimana tidak? Bahasa-bahasa *scripting* tersebut ditulis dengan menggunakan bahasa C.

Apabila bahasa lain menyerah pada pemrograman yang melibatkan akses ke hardware dan kernel sistem operasi, bahasa ini malah digunakan untuk menulis sistem operasi dan berbagai rutin penting sistem operasi tersebut. Kernel Linux juga ditulis menggunakan bahasa ini. Belum lagi sistem operasi lainnya seperti berbagai varian UNIX dan Microsoft Windows.

Bahasa C adalah bahasa yang sangat kuat. Anda hampir bisa melakukan apa saja dengannya, termasuk melakukan pemrograman web. Protokol-protokol jaringan juga umumnya diimplementasikan menggunakan bahasa C.

Bagaimana dengan performa? Hanya satu jawaban singkat yang dapat diberikan: Bahasa C hanya kalah dari *assembly*. Tapi walau begitu, *assembly* tidak memiliki kemampuan portabilitas tinggi seperti yang dimiliki oleh bahasa

C. Dibandingkan dengan Python dan Java? Kecepatan eksekusi aplikasi yang ditulis dengan menggunakan kedua bahasa terakhir sudah boleh diacungkan jempol apabila mampu mencapai 1/5 dari kecepatan eksekusi aplikasi yang ditulis menggunakan bahasa C. Umumnya, aplikasi yang ditulis dengan Python ataupun Java berkisar antara 5 sampai 10 kali lebih lambat.

Tapi tunggu dulu, karena bahasa C juga memiliki kekurangan yang cukup besar. Waktu pengembangan! Jangan harap membuat aplikasi secara cepat menggunakan bahasa ini. Anda harus menghadapi berbagai problem yang disebabkan oleh penggunaan pointer, seperti *buffer overflow*, pengolahan *string*, dan berbagai hal lainnya. Belum lagi jika Anda harus menggunakan *regular expression* di dalam aplikasi Anda. Walau telah dibantu oleh berbagai pustaka tambahan yang ada, tetap saja waktu pengembangan yang diperlukan masih cukup besar. Belum lagi permasalahan yang mulai timbul apabila skala aplikasi Anda membesar. Apabila program yang ditulis menggunakan Python berjalan lebih lambat 5 sampai 10 kali, waktu pengembangan yang dibutuhkan oleh Python juga lebih sedikit 5 sampai 10 kali. Kekurangan lainnya adalah bahasa C relatif lebih susah dipelajari oleh pemula di bidang pemrograman. Jika permasalahan Anda berada pada aplikasi yang langsung berhadapan dengan *end user* dan tidak memerlukan kecepatan yang luar biasa, letakkan C pada urutan terakhir bahasa yang akan digunakan. Untuk mempercepat waktu pengembangan Anda, *tool* seperti *autoproject* dapat digunakan.

Apabila aplikasi yang akan dibangun langsung berhubungan dengan kebutuhan end user sehari-hari dan dapat menjadi sangat kompleks,

sementara cinta Anda kepada bahasa C sudah tidak dapat dipisahkan lagi, cobalah beralih untuk menggunakan C++. Yang terakhir ini berorientasi pada objek dan sangat tepat digunakan untuk aplikasi yang kompleks.

G++ pada GCC yang digunakan untuk melakukan kompilasi program C++ sendiri dapat diandalkan. G++ bukanlah sebuah preproesor, melainkan sebuah kompiler. G++ membangun *object code* dari *source code* program C++ secara langsung. Hal ini bertolak belakang dengan beberapa implementasi yang membuat *source code* program C terlebih dahulu dari *source code* program C++.

Perlu untuk membangun aplikasi yang indah dan *user friendly*, namun tetap memiliki performa tinggi dengan C/C++? Gunakan saja berbagai GUI toolkit yang ada di pasaran open source. Sebut saja GTK+ dan QT. Dengan menggabungkan kemampuan tinggi bahasa C/C++ dan keindahan GUI toolkit tersebut, sebuah aplikasi yang indah, user friendly namun memiliki performa tinggi dapat dicapai.

Sebagai studi kasus, Anda dapat melirik GNOME yang dibangun dengan C dan GTK+, serta KDE yang dibangun dengan C++ dan QT.

Merujuk kepada standar ANSI dan ISO bahasa C, Anda dapat menggunakan opsi `-ansi`, `-std=c89` atau `-std=iso9899:1990` pada saat kompilasi untuk menulis aplikasi dalam bahasa C standar ANSI dan ISO. GCC tidak mendukung standarisasi ISO 1990 yang memiliki kesalahan.



Di dunia Linux, seringkali pula terdengar istilah EGCS. Ada pula yang menghubungkannya dengan GCC. Sebenarnya, ada apa dengan EGCS dan GCC? Mari kembali ke zaman sekitar 12 tahun yang lalu. Pada tahun 1990/1991, GCC versi 1 telah mencapai tingkat stabilitas yang memadai kalau tidak mau dikatakan tinggi. Untuk setiap target yang didukung oleh GCC, kita dapat membangun aplikasi yang akan bekerja dengan baik. Walau demikian, tentunya

GCC pun perlu dikembangkan lagi. Dan karena desain GCC 1 terlalu mengikat, maka pengembangannya akan difokuskan para perubahan mayor yang akan menghasilkan GCC versi 2.

Kemudian, GCC 2 pun dikembangkan dan pada saat pengembangan tersebut telah membawa GCC ke tahap "dapat digunakan", pengembangan GCC 1 pun dihentikan dan semua anggota tim mengarahkan usahanya untuk membuat GCC bekerja sebaik mungkin, jauh melebihi GCC 1. Langkah ini berjalan harmonis dengan proyek EGCS yang berdiri pada tahun 1997.

Dan pada bulan April 1999, Free Software Foundation secara resmi menghentikan pengembangan GCC 2 dan menunjuk proyek EGCS sebagai *maintainer* resmi dari GCC. Hasilnya adalah sebuah proyek yang terus mengembangkan GCC dengan kontrol dari GCC Steering Committee, yang didirikan pada tahun 1998 dan diresmikan oleh FSF pada bulan April 1999.

Seperti yang telah dibahas bahwa GCC tidaklah dikontrol oleh RedHat, berikut ini adalah para anggota Steering Committee tersebut: **Per Bothner** (Brainfood Inc), **Joe Buck** (Synopsys), **David Edelsohn** (IBM), **Kaveh R. Ghazi**, **Torbjorn Granlund** (Swox AB), **Jeffrey A. Law** (Red Hat), **Marc Lehmann** (Technische Universität Karlsruhe), **Jason Merrill** (Red Hat), **David Miller** (Red Hat), **Mark Mitchell** (CodeSourcery), **Toon Moene** (Koninklijk Nederlands Meteorologisch Instituut), **Gerald Pfeifer** (Vienna University of Technology), **Joel Sherrill** (OAR Corporation), **Jim Wilson**.

Compiler adalah alat yang sangat penting dalam pengembangan *software*. Sebuah compiler pun akan sangat berpegang pada patokan spesifikasi suatu bahasa program. Umumnya, spesifikasi suatu bahasa program tidaklah sering berubah (spesifikasi bahasa Perl 6 berubah setelah digunakan lebih dari 10 tahun).

Akan tetapi beda dengan suatu kompiler. Penggunaan *resource* secara lebih efisien dan penambahan *feature*

adalah beberapa alasan perubahan versi compiler. Tentunya terlepas dari perbaikan *bugs*. Dan apa yang terjadi apabila suatu kompiler berubah? Bisa-bisa suatu *source code* program tidak dapat dikompilasi dengan versi baru dari kompiler tersebut. Dan tentunya sangat mungkin apabila sebuah *source code* program tidak dapat di-compile dengan versi lama dari kompiler tersebut.

Dan bagaimana kalau Anda memerlukan kedua macam program tersebut dan Anda harus melakukan kompilasi terhadap keduanya? Bagaimana pula kalau Anda adalah seorang *maintainer* paket suatu distro yang harus melakukan kompilasi untuk dua versi program karena distro Anda menerapkan sistem rilis *stable* dan *unstable*? Atau bagaimana kalau Anda harus melakukan *backport* dari suatu sistem *unstable* ke sistem *stable*? Jawabannya bisa saja dengan menginstalasi dua versi compiler. Menginstalasi dua versi GCC.

Anda bisa menggunakan salah satu trik berikut ini jika harus menginstalasi dua versi GCC dalam satu sistem. Pertama dan paling konvensional adalah dengan melakukan instalasi dengan *prefix* yang berbeda. Tapi untuk yang satu ini, Anda perlu membuat *symlink*. Sedikit repot, akan tetapi sistem Anda akan relatif bersih. Cara yang kedua, Anda harus bekerja cukup keras. Cara yang memodifikasi—program—*transform-name* ini sebaiknya dihindari kecuali Anda tahu persis apa yang sedang Anda lakukan.

Dan cara yang paling mudah adalah dengan memodifikasi—program—*suffix* pada saat menjalankan *configure*. Cara yang satu ini akan membuat salah satu versi gcc Anda memiliki nama file yang berbeda.

Berikut ini adalah contoh kode untuk mencetak tulisan Hello World:

```
#include <stdio.h>

int main(void){

    printf ("Hello World\n");

    return 0;
}
```

Java



Kalau ada suatu bahasa yang sangat fenomenal yang dapat mengubah cara komputasi dan matang untuk masa depan, maka itulah

Java. Predikat tersebut rasa-rasanya tidak berlebihan. Kenapa? Dari hal yang paling sederhana saja, Anda bisa menemukan berbagai java applet ketika menjelajahi Internet. Anda juga bisa menemukan berbagai aplikasi java di *desktop*. Kemudian ponsel Anda pun barangkali mendukung Java.

Lalu, ada game yang ditulis menggunakan Java. Kemudian peralatan rumah tangga. Itu masih belum terhitung aplikasi untuk *server*. Dan entah apa lagi yang tiba-tiba dapat menjalankan bahasa pemrograman yang memiliki logo dengan nama Duke tersebut. Java sendiri bukan hanya sebuah bahasa pemrograman. Java juga merupakan *platform* dan arsitektur untuk komputasi jaringan.

Mari sejenak melongok ke-12 tahun yang silam, tahun 1991, ketika **James Gosling**, pencetus Java, mulai merintis suatu bahasa pemrograman baru. Waktu itu, proyek tersebut masih bernama *Stealth Project*. Di dalam proyek tersebut, James adalah personel yang bertanggung jawab akan bahasa pemrograman. Bahasa pemrograman yang sedang ditulis olehnya haruslah memiliki sebuah nama. Lalu tercetuslah "oak" yang diambil dari nama pohon yang kelihatan di jendela manusia jenius tersebut.

Perkembangan Java pun tidak terlepas dari perkembangan Internet. Keduanya adalah pasangan setia yang telah mengalami susah senang selama bertahun-tahun. Kelahiran NCSA Mosaic, *web browser* grafikal pertama semakin memacu perkembangan "oak" tersebut. Ketika "oak" telah semakin mantap, salah satu personel tim manajemen Sun, terbang sejauh 300000 mil untuk memperkenalkan teknologi "oak" kepada perusahaan-

perusahaan elektronik dan konsumen lainnya yang potensial menggunakan teknologi tersebut.

Arthur Van Hoff, jenius lainnya juga merupakan tokoh kunci dalam pengembangan Java. Beliau termasuk salah satu orang yang memantapkan konsep Java. Van Hoff pula yang mengimplementasikan Java menggunakan Java. Sementara James mengimplementasikan Java menggunakan bahasa C.

Sebelum Java menjadi populer dan dirilis untuk masyarakat umum, tim pengembang Java juga mengembangkan suatu web browser bernama *WebRunner* yang kemudian menjadi *Hot Java*. Sepertinya kehadiran Internet dan kedekatan tim pengembang Java dengan orang-orang di *Netscape* juga semakin memacu perkembangannya. Tim *Netscape* dan tim pengembang Java adalah pencetus teknologi yang akan sangat besar di masa mendatang.

Apabila kita ingin mendeskripsikan Java, barangkali inilah yang akan terucap: suatu teknologi yang sederhana, berorientasi objek, terdistribusi, intepreter, luar biasa, aman, netral terhadap arsitektur, portabel, memiliki performa tinggi, *multithreaded*, dan dinamis. Ya, sepertinya itu sudah cukup memborong hal-hal yang baik-baik untuk suatu teknologi.

Bicara soal dukungan platform, Java sudah tidak diragukan lagi. Sebut saja platform Anda dan hampir dipastikan Java bisa berjalan di sana. Teknologi *Java Virtual Machine* membuat java semakin portabel. Bicara soal dukungan platform mungkin akan membuang waktu. Dan jika Anda bertanya kalau Java begitu hebat, apakah kekurangan Java? Apa yang tidak mampu dilakukan oleh Java?

Modul keamanan Java hanya dapat diaplikasikan kepada program yang ditulis menggunakan Java. Walau demikian, Java tetaplah bahasa pemrograman pertama yang menyertakan konsep keamanan terintegrasi di Java sendiri.

Akan tetapi, bagaimanapun, karena semua program Java dijalankan di dalam

runtime environment, program atau *thread* di luar environment ini tidak dapat dikontrol oleh *framework* keamanan Java. Mekanisme keamanan Java hanya melindungi sumber daya lokal. Proteksi sumber daya melibatkan Java *Virtual Machine* dan karena setiap runtime environment Java dijalankan di komputer tertentu, maka proteksi hanya dapat ditujukan kepada mesin tersebut.

Pada akhirnya, konsep keamanan pada Java tidak dapat mencegah tindakan-tindakan merugikan jika pengguna salah melakukan konfigurasi. Hal ini dirasakan cukup wajar karena pengguna adalah faktor terbesar dalam kegagalan keamanan komputer. Salah satu "kekurangan" Java lainnya adalah lamanya waktu eksekusi. Dibandingkan *Python*, Java masih relatif lebih lambat. Dan sebaiknya jangan coba-coba bandingkan waktu eksekusi program Java dengan waktu eksekusi program C. Anda akan kaget dan kemudian kecewa.

Java memang begitu menarik. Walau demikian, Anda perlu sedikit latar belakang pemrograman untuk mempelajari bahasa yang satu ini. Java adalah bahasa pemrograman yang berorientasi objek. Oleh karena itu, pemahaman konsep objek haruslah kuat sehingga tidak mengganggu perancangan aplikasi.

Untungnya, Java telah dilengkapi *RAD* yang cukup baik, yang sekarang ini menjadi bagian integral dari Java *ONE (Open Net Environment)*. Sedikit menyinggung yang satu ini, beberapa pendapat (yang rasa-rasanya cukup objektif, dari pemilihan teknologi dan potensi skalabilitas) mengatakan Java *ONE* lebih unggul dari teknologi *.NET* Microsoft.

Membuat program berbasis GUI? Java sangat bisa diandalkan. Contohnya adalah applet yang beredar luas di Internet. Pembuatan GUI di Java umumnya melibatkan *Abstract Window Toolkit (AWT)* dan *Swing*. Umumnya itu saja sudah cukup.

Pertanyaan berikutnya: sejauh mana Java dapat bekerja sama dengan bahasa pemrograman lain? Kemudian



▲ James Gosling, inovator Java

bagaimana dengan Java di Linux? Bicara soal kerja sama dengan bahasa lain, Java dapat diandalkan. Kerja sama dengan Python akan memberikan kombinasi stabilitas dan cepatnya waktu pengembangan aplikasi. Para *hacker*

Linux pun semakin meningkatkan dukungan akan Java. Segera dapatkan Java di <http://java.sun.com>.

Ada terselip rasa bangga ketika mendengar nama bahasa pemrograman yang satu ini. Ya, bagaimana tidak kalau nama tersebut sama seperti nama salah satu pulau besar di Indonesia? Dan bagaimana tidak bangga kalau beberapa proyek yang berkaitan dengannya juga menggunakan istilah-istilah yang umum di Indonesia? Salah satunya adalah Batik.

Sebenarnya, darimanakah asal kata Java? Nama tersebut adalah hasil dari banyak pemikiran dan diskusi oleh para pengembang Java. Para jenius tersebut ingin membuat Java sebagai bahasa yang mencerminkan kehidupan, animasi, kecepatan, interaktif, dan hal-hal lainnya. Sebagai akhirnya, kata Java pun terpilih sebagai pemenang dari sekian banyak pilihan. Java sendiri bukanlah singkatan. Kalau Anda ingat Java, ingatlah selalu kehangatan dan khasiat aromatiknyanya.

Kalau berkunjung ke situs java.sun.com, Anda akan menemukan logo Java yang lucu. Akan tetapi, apakah logo tersebut bebas dipakai oleh siapa saja? Siapa yang memegang hak kepemilikan logo tersebut? Anda dapat mengunjungi http://java.sun.com/nav/business/trademark_guidelines.html untuk keterangan selengkapnya apabila Anda ingin menggunakan logo Java.

Kalau diperhatikan, sedari awal kita membahas Java, nama Sun Microsystems selalu terlintas. Sun sendiri adalah inovator Java. Akan tetapi di luar sana, Anda juga dapat menemukan versi lain dari Java yang bekerja mengikuti spesifikasi bahasa

pemrograman Java. Distro seperti Debian yang sangat memegang teguh Debian Free Software *guidelines*-nya tidak memasukkan Java dari Sun Microsystems.

Akan tetapi, Anda tetap dapat menggunakan Java yang berasal dari Blackdown. Blackdown Java dapat menjalankan (hampir) semua aplikasi Java yang ditulis menggunakan Java dari Sun.

Sun identik dengan perkembangan Internet, sementara Internet dan web browser adalah teman akrab. Tidak dapat dipisahkan, begitulah adanya. Kalau dalam sejarah Java di Sun melibatkan kelahiran Hot Java, lantas bagaimana dengan nasib web browser yang satu tersebut saat ini? Hot Java telah mencapai versi 1.1Beta1. Anda dapat mengunjungi <http://java.sun.com/products/hotjava> untuk informasi lebih lengkapnya.

Dan bagi Anda yang penasaran untuk mendapatkan *source code* Java dari Sun, arahkanlah web browser Anda ke alamat <http://java.sun.com/communitysource/index.html> Anda bisa mendapatkan *source code* bahasa Java. Walau demikian, Sun tidak menyertakan class `sun.*` sehingga Anda tidak dapat membangun Java secara komplit dengan mengandalkan *source* tersebut. Gunakan hanya untuk keperluan pembelajaran dan sebagai referensi cara kerja teknologi Java.

Salah satu hal terbaik yang datang bersama bahasa ini adalah dokumentasi. Sampai saat ini, rasa-rasanya susah bagi bahasa pemrograman lain untuk dapat menyamai dokumentasi Java. Walaupun Python telah datang bersama `pydoc` dan Perl dengan `CPAN`nya.

Ya, waktu pengembangan aplikasi Java cukuplah lama. Untuk mencetak sebuah Hello World saja, Anda harus menuliskan kode seperti ini:

```
import java.*;

class hello{
    public static void main(String[] args){
        System.out.println("Hello World");
    }
}
```

Bahasa lain-lain

Kalau mau jujur, dunia *open source* sedang kelebihan program. Terlepas dari tidak meratanya kelebihan tersebut (sebagai contoh, ada ratusan teks editor sementara *word processor* yang baik dapat dihitung dengan jari tangan), bahasa pemrograman termasuk salah satu jenis program yang melimpah ruah. Bagaimana tidak? Dari pertengahan tahun 80 an sampai 90 an saja, bahasa-bahasa baru bermunculan. Sekarang pun makin bermunculan *binding-binding* untuk bahasa-bahasa tersebut. Satu fenomena yang susah dicari di dunia *proprietary*.

Kalau Anda bertanya soal bahasa apa yang harus dipelajari dan kami harus menjawab, maka hal tersebut akan kembali ke diri masing-masing. Apa yang menjadi kebutuhan dan bagaimana tingkat skalabilitasnya? Tingkat skalabilitas dibutuhkan karena tidak semua bahasa dirancang untuk mampu memiliki tingkat skalabilitas yang tinggi. Contoh paling baik adalah Java. Python segera menyusul. Apabila suatu hari aplikasi yang telah Anda rancang harus berjalan dalam sistem yang lebih skalabel, pastikan Anda tidak perlu menulis ulang semua kode tersebut. Secara mendasar, untuk pemenuhan kebutuhan saja, banyak sekali bahasa pemrograman yang bisa digunakan. Sebagai contoh, untuk melakukan pemrograman web yang baik misalnya, Anda bisa menggunakan Java, Python, PHP, Perl, dan masih banyak lagi. Semuanya menawarkan pemecahan masalah. Semuanya mampu melakukan hal mendasar untuk pemrograman web, seperti koneksi ke database *server* misalnya.

Jadi, kembali kepada Anda dan kebutuhan. Soal populer atau tidak, soal mudah mencari peluang kerja atau tidak, rasa-rasanya itu banyak kembali ke diri masing-masing. Kalau Anda bisa membuat sebuah aplikasi yang dibutuhkan oleh banyak orang menggunakan bahasa yang kurang populer, lantas siapa yang peduli kalau Anda menggunakan bahasa tersebut?

Perl

USE OF THE LOGO



Ini dia bahasa yang disebut-sebut sebagai pasangan setia dari sistem operasi Linux. *Practical Extraction and Report Language* adalah salah satu bahasa tua dan tetap mengklaim dirinya sebagai bahasa *post modernisme*. Belasan tahun yang lalu, tepatnya pada tahun 1987, **Larry Wall**, seorang *hacker* kawakan menulis bahasa yang satu ini. Sesuai namanya, bahasa ini ditujukan untuk mempermudah banyak hal dibandingkan penggunaan C/C++ . Sampai saat ini, tidak dapat dipungkiri lagi, interpreter yang satu ini telah menjadi bahasa yang sangat favorit, apalagi dikalangan para veteran sistem operasi UNIX dan Linux.

Telah dibahas bahwa Perl sangat cocok apabila dipadankan dengan Linux. Namun, Perl tidak hanya saling setia dengan Linux. Perl dapat pula digunakan di Mac dan Windows. Bahkan tercatat dalam sejarah, Perl untuk Mac (versi 4.0.2) telah dirilis pada bulan Januari 1992 oleh **Matthias Neeracher**. Bagi Anda yang bermain-main di sistem *embedded*, *perlembded* pasti menarik perhatian Anda. Dan bicara soal lisensi, Anda pun tidak perlu mengkhawatirkan hal yang satu ini. Lisensi Artistic Perl membuat Anda bebas menggunakan Perl sesuai kebutuhan Anda, baik sekadar hobi ataupun untuk keperluan bisnis.

Salah satu hal yang cukup menarik di hutan belantara Perl adalah slogan-slogan yang dilahirkan. Tidak hanya Larry Wall, akan tetapi **Randal L. Schwartz** dan **Tom Christiansen** yang terhitung sebagai para *developer* utama pun sangat senang menciptakan berbagai slogan. Sebut saja Just another Perl Hacker oleh Randal. Bagi yang menyenangi Larry Wall, Anda bahkan dapat membaca slogan-slogan yang diciptakannya di <http://www.perl.com/CPAN/misc/lwall-quotes.txt.gz>.

Bagi seorang pemula dalam pemrograman, Perl juga bukan termasuk bahasa yang sulit dipelajari. Kedekatannya

dengan bahasa manusia pun sangatlah tinggi. Walau sayangnya, dan hal ini pun diakui oleh beberapa anggota milis Perl, penggunaan *regular expression* pada bahasa yang satu ini bisa-bisa membuat takut pemula yang ingin mempelajari Perl.

Akan tetapi secara umum, jika Anda adalah seorang pemula dan menginginkan pemrograman yang cepat dan ringkas, bisa mencoba Perl. Bagi Anda pula yang kurang menyukai keseragaman aksi untuk suatu tujuan tertentu, slogan berikut ini mungkin akan menarik perhatian Anda: *"There's more than one way to do it"*. Ya, umumnya, dalam menggunakan Perl, kita dapat menggunakan berbagai cara untuk menyelesaikan satu permasalahan. Hal ini berlawanan dengan Python yang lebih mementingkan keseragaman cara.

Di luar semua hal tersebut, daerah kekuasaan Perl sangatlah luas. Sebut saja mulai dari administrasi sistem, pembuatan aplikasi untuk *desktop* berbasis GUI, aplikasi web dan server-server. Dalam hal daerah kekuasaan, yang satu ini pantas diacungi jempol. Beberapa veteran aplikasi web mungkin akan menunjuk Perl sebagai pilihan untuk pembuatan aplikasi CGI. Bahkan PHP pun, yang notabene semakin terkenal sebagai bahasa untuk web pun mencontek sintaks dari Perl. Daerah kekuasaan Perl pun semakin meluas dengan tersedianya modul-modul untuk Perl.

Perl memiliki sangat banyak modul yang akan memudahkan Anda untuk bekerja dengan hal-hal yang tidak lazim sekalipun. Walau demikian, sebaiknya, untuk tugas-tugas berat seperti aplikasi yang mementingkan *context switching*, *low-level* sistem operasi, aplikasi *multi-threaded shared-memory* kompleks, serta sistem *embedded real-time*, Anda tidak memilih Perl.

Untuk pemrograman dengan skala lebih besar dan berorientasi objek, Anda juga masih bisa menggunakan Perl. Walau dalam banyak hal, pemrograman terstruktur untuk menyelesaikan masalah tertentu juga sangat lazim ditemui.

Anda berorientasi ke pemrograman berbasis GUI dan mencari RAD yang baik untuk Perl? Hal tersebut dapat dibilang kurang lazim. Umumnya, Anda sudah



▲ **Larry Wall**, inovator perl

cukup dimanjakan oleh binding berbagai GUI toolkit yang ada. Apabila Anda cukup familiar dengan TK, pembuatan aplikasi GUI dengan Perl akan terasa menyenangkan. Umumnya, hampir semua GUI toolkit memiliki binding untuk Perl.

Aplikasi-aplikasi lain seperti XChat dan GIMP pun dapat dikembangkan lebih luas dengan bantuan *scripting* Perl. Web server Apache pun mencatatkan dirinya sebagai pengguna Perl yang cukup intensif.

Bagaimana perbandingan antara Perl dengan bahasa lain? Bagi Anda yang peduli, artikel di situs <http://language.perl.com/versus/> mungkin akan terasa cukup menarik. Anda akan menemukan perbandingan dengan Java, Python, REXX, Tcl, dan lain sebagainya. Kunjungi pula <http://www.cpan.org> untuk mendapatkan *resource-resource* tentang Perl.

Artikel ini sempat menyinggung Perl sebagai bahasa *post modernisme*, yang diungkapkan oleh Larry Wall. Zaman *modernisme* lebih mementingkan logika OR daripada logika AND. Sementara zaman *post modernisme* lebih mementingkan kebalikannya. Logika AND memiliki tingkat kepentingan yang lebih tinggi. Di Perl, AND memiliki preseden yang lebih tinggi dibandingkan OR. Hal tersebut membuktikan Perl sebagai bahasa *post modernisme*.

Kalau programmer adalah seniman, maka Perl adalah salah satu instrumen yang sangat mendukung hasil karya seni dari seniman tersebut. Termasuk hasil karya berupa kata-kata indah yang dicerminkan pada alamat situs-situs berikut: perl.com, pm.org, perl.org, dan cpan.org. Situs-situs tersebut pun terhitung resmi. Yang tidak resmi sendiri masih sangat banyak. Kunjungi cpan.org untuk mengakses arsip raksasa Perl!

Tertarik untuk melihat sekilas sintaks Perl? Berikut ini adalah contoh program "Hello World":

```
#!/usr/bin/perl
print "Hello World\n";
```



PHP



Adalah **Rasmus Lerdorf** yang pada tahun 1995 tertarik untuk

mengetahui siapa saja yang telah mengakses resume dirinya yang diletakkan secara *online*. Beliau pun menulis beberapa *script* Perl untuk itu. Beberapa *script* Perl tersebut dinamakan sebagai Personal Home Page Tools.

Akan tetapi, rupa-rupanya *script-script* tersebut masih belum bisa memenuhi kebutuhannya. Beliau pun menuliskan kembali fungsi-fungsi yang dibutuhkan dalam bahasa C. Dan yang terakhir ini mampu melakukan tugas-tugas yang lebih berat seperti berkomunikasi dengan database *server* dan melakukan pemrograman dinamis untuk halaman web. Dan Rasmus pun kemudian merilis program yang telah ditulisnya tersebut dengan nama PHP/FI, di mana FI adalah singkatan dari Form Interpreter.

Pada tahun 1997, PHP/FI 2.0 pun dirilis setelah mengalami penulisan ulang dalam bahasa C. Kontribusi pun datang dari berbagai pelosok. Sebagai hasilnya, dilaporkan pengguna PHP pun meningkat menjadi 50.000 domain atau sekitar 1% dari domain di Internet waktu itu.

PHP/FI sendiri lebih merupakan proyek pribadi Rasmus daripada proyek yang besar. Kemudian datanglah **Andi Gutmans** dan **Zeev Suraski** yang menulis ulang inti dari PHP, yang merupakan cikal bakal PHP saat ini. Dan sebagai hasilnya, pada bulan November 1998, PHP 3.0 pun dirilis dan secara resmi menggantikan kedudukan PHP/FI 2.0. Banyak perubahan yang terjadi, termasuk perubahan kepanjangan PHP menjadi *PHP Hypertext Preprocessor*. Pengguna pun semakin bertambah sampai sekitar 10% dari *web server* yang ada di Internet.

Kepiawaian Andi dan Zeev pula yang membuat lahirnya PHP 4.

Mereka juga memperkenalkan Zend Engine, yang merupakan singkatan dari nama Zeev dan Andi. Zend Engine ini merupakan salah satu inti dari PHP. Saat ini, PHP 4 telah digunakan secara amat meluas di Internet. PHP sendiri dilisensikan di bawah bendera PHP License. Anda bisa mendapatkan PHP di <http://www.php.net>.

Bicara soal dukungan *platform* lain, bahasa yang satu ini juga pantas diberikan acungan jempol. Minimal, jika Anda menggunakan Windows ataupun Linux, bahasa yang satu ini sudah dapat dipergunakan dengan mulus. Dan di luar masalah *platform*, PHP adalah jago di bidang koneksi database. Sebut saja database yang Anda unggulkan dan hampir bisa dipastikan Anda bisa bekerja langsung dengan database tersebut. Dukungan akan database memang merupakan salah satu fitur utama dari PHP.

Sama seperti halnya Python, PHP juga bukan merupakan bahasa yang rumit. Walau PHP memiliki sintaks serupa dengan C dan Perl, Anda tidak perlu menjadi veteran kedua bahasa tersebut untuk mempelajarinya. Cukup gunakan manual yang dapat di-download di Internet atau membeli buku panduan PHP dan sebuah situs nandinamis pun akan meluncur keluar dari tangan Anda dengan mulus.

Ya, bahasa ini memang memiliki daerah kekuasaan di bidang pemrograman web. Walau sejalan dengan perkembangan zaman, PHP juga dapat digunakan bersama dengan GTK+ untuk membangun aplikasi desktop. Jika Anda seorang fans setia PHP dan ingin membangun aplikasi berbasis GUI, gunakan saja PHP-GTK.

Nah, kini kita sampai pada urusan penting setiap bahasa pemrograman: performa. Tentunya, kita sangat tidak berharap untuk menemukan dan membuka situs yang sangat lambat, yang dibangun dengan PHP. Beberapa artikel di internet, termasuk artikel yang ditulis oleh para pengembang PHP membuktikan hal tersebut. Bahkan, dari hasil perbandingan antara

PHP dengan bahasa lain seperti ASP (sebenarnya ASP bukanlah sebuah bahasa, karena ASP menggunakan VBScript) menunjukkan PHP unggul dari sisi *free*, dukungan web server, kecepatan dan kestabilan. Dibandingkan dengan Cold Fusion, PHP kalah di dalam hal penanganan kesalahan, abstraksi database dan kemudahan. Dan apabila dibandingkan dengan Perl, PHP sendiri masih kurang fleksibel.

Bicara soal RAD dan pemrograman berorientasi objek, PHP rasa-rasanya masih ketinggalan dengan bahasa lain. PHP sangat mendukung pemrograman berorientasi objek, akan tetapi, sama seperti Python, Anda tidak harus menulis sebuah *class* secara eksplisit hanya untuk pembuatan aplikasi sederhana. Kebiasaan tidak harus berorientasi pada objek ini terkadang membuat kode program yang ditulis menjadi sulit dibaca.

PHP sendiri telah dipakai secara sangat meluas di Internet. Sebut saja 100 situs dan tidak heran apabila Anda dapat menemukan hampir 70 atau 80 situs yang menggunakan PHP. *Web hosting* yang mendukung PHP pun sudah tidak terkatakan banyaknya, sehingga akan sangat memudahkan apabila publikasi akan dilakukan.

Dengan semakin berkembangnya PHP, proyek-proyek besar pun bermunculan. Dari sisi *content management* saja, Anda bisa menemukan PHPNuke dan PostNuke. Ada pula PHPGroupware untuk urusan pekerjaan kolaboratif. Hanya dengan menggunakan proyek-proyek tersebut, Anda bahkan dapat menghasilkan situs yang luar biasa tanpa banyak melakukan penulisan kode program. Selain itu, urusan instalasi pun tidak rumit. Apabila Anda menggunakan web server Apache dan database server MySQL, PHP akan bersikap sangat manis di sini.

Bahasa-bahasa pemrograman lain yang kita bahas dapat digunakan secara mudah untuk mengakses system. Lantas, bagaimana dengan



« Rasmus Lerdorf, inovator PHP

PHP? Jika Anda adalah seorang programer sistem yang berkepentingan

untuk menampilkan informasi sistem Anda di web, Anda masih bisa menggunakan PHP. Kedekatan sintaks PHP dengan Perl, C, dan shell menyebabkan Anda yang terbiasa dengan bahasa-bahasa tersebut relatif lebih mudah untuk bekerja dengan sistem. Walau demikian, kemampuan tersebut tetap bisa dibatasi mengingat akses langsung pada sistem dengan aplikasi web adalah hal yang cukup riskan.

Masih bicara soal fitur yang berguna langsung bagi dunia nyata, PHP cukup andal. Mau membangun *groupware*? Dapatkan contohnya dengan PHPGroupware. Kemudian ada pula LDAPexplorer yang berguna untuk mengakses LDAP. Untuk Anda yang bekerja dengan komunitas, dapat menggunakan PHPNuke ataupun PostNuke. Melengkapi daftar tersebut, kunjungilah <http://twig.screwdriver.net/> untuk mendapatkan contoh-contoh bagaimana menggantikan Microsoft Exchange. Luar biasa!

Salah satu kunci sukses PHP adalah dukungan untuk teknologi terbaru dan dukungan pustaka-pustaka populer lainnya. Sebut saja database server besar yang tidak didukung oleh PHP! Boleh dikatakan tidak ada! PHP juga dapat digunakan untuk membuat *image secara on the fly di internet*. Bahkan Anda dapat langsung membuat PDF langsung dengan PHP. Anda yang bekerja dengan komputer besar macam AS400 dan database luar biasa DB/2 pun tetap dapat menggunakan PHP. Dapatkan keterangannya di <http://www.php-faq.com/as400.php>.

Mungkin pada saat PHP ditulis, Rasmus sendiri tidak akan pernah membayangkan bahasa ini akan melesit dan begitu terkenal di Internet. Kursus-kursus pemrograman web seperti ini tidak bisa dikatakan lengkap apabila

belum memasukkan materi yang satu ini. Tutorial di Internet pun berlimpahan bak banjir. Bagi Anda yang masih baru, beberapa alamat situs di bawah ini dapat dijadikan bahan referensi:

- ➔ <http://www.rci.rutgers.edu/~jfulton/php1/index.html>
- ➔ <http://www.newbienetwork.net/>
- ➔ <http://www.thickbook.com/>
- ➔ <http://www.phpdeveloper.org/>
- ➔ <http://www.weberdev.org/>
- ➔ <http://www.weberdev.org/>
- ➔ <http://www.devshed.com/>
- ➔ <http://www.webmonkey.com/>
- ➔ <http://www.devnetwork.net>
- ➔ <http://www.irc-html.com/>
- ➔ <http://www.phpbeginner.com>
- ➔ <http://www.phpbuilder.com/>
- ➔ <http://www.zend.com/>

Untuk kebutuhan aplikasi web, PHP memerlukan sebuah *web server* untuk dapat bekerja. Umumnya, Apache saja sudah sangat mencukupi. Dan jika Anda ingin meletakkan program PHP di Internet, Anda akan memerlukan *web hosting* yang mendukung PHP. Apabila hendak mencoba-coba saja, tentunya Anda masih bisa menggunakan web hosting gratisan. Untuk kebutuhan yang serius, beberapa web hosting berikut ini telah terbukti andal sebagai web hosting PHP:

- ➔ <http://www.connecticut-web.com/>
- ➔ <http://www.cedant.com/>
- ➔ <http://www.spyproductions.com/>
- ➔ http://site-works.com/hosting_plans_comparison_chart.html
- ➔ <http://www.aletiahosting.com/>
- ➔ <http://dsvr.co.uk/>
- ➔ <http://www.datasnke.co.uk/>
- ➔ <http://www.nomonthlyfees.com/>
- ➔ <http://www.weberdev.com/index.php3?GoTo=phenominet/prices.htm>
- ➔ <http://www.phpwebhosting.com/>
- ➔ <http://www.cobaltconnection.com/>
- ➔ <http://www.pair.net/>
- ➔ <http://www.rackspace.net/>

Untuk daftar selengkapnya, Anda dapat mengunjungi alamat <http://www.zend.com/links/>

[links.php?CID=62](http://www.zend.com/links.php?CID=62). Jangan lupa mengunjungi <http://www.webhostingtalk.com> untuk mendapatkan perbandingan dan *review* untuk perusahaan-perusahaan web hosting. Beberapa web hosting di Indonesia seperti Masterwebnet dan Cakraweb juga dapat Anda andalkan. Beberapa web hosting bahkan menerapkan konsep *one price solution*.

Di luar semua itu, seiring dengan meningkatnya penggunaan Internet dan e-mail, kebutuhan webmail sebagai sarana akses e-mail paling praktis pun semakin bertambah. Dan apabila Anda adalah penyedia layanan Internet, Anda tidak perlu menulis sendiri aplikasi webmail, karena beberapa webmail seperti Squirrelmail dan Basilix sangat dapat diandalkan. Dan masih berhubungan dengan komunitas, beberapa *software bulletin board* berikut ini juga ditulis dengan PHP:

- **YaBB** (<http://www.yabbse.org/>)
- **phpBB** (<http://www.phpbb.com/>)
- **Phorum** (<http://www.phorum.org/>)
- **fudforum** (<http://fud.prohost.org/>)
- **TikiWiki** (<http://tikiwiki.sourceforge.net/>)

Masih kurang? Baiklah, kita akan berpindah ke aplikasi lainnya. MP3 Jukebox misalnya. Ada pula yang dibangun dengan PHP. Silakan arahkan web browser Anda ke <http://netjuke.sourceforge.net/>. Contoh lain aplikasi siap pakai seperti *shopping cart* yang dibangun dengan PHP bisa Anda dapatkan di:

- ➔ **Fish Cart** (<http://www.fishcart.org/>)
- ➔ **OsCommerce** (<http://www.oscommerce.com/>)
- ➔ **Phpshop** (<http://www.phpshop.org/>)

PHP termasuk *server side scripting* dan disimpan di server. Berikut ini adalah kode untuk mencetak tulisan "Hello World" ke web browser:

```
<?
print "Hello World";
?>
```

Python



Sekilas, nama bahasa pemrograman yang satu ini terdengar seperti nama salah satu reptil buas. Akan tetapi, Anda tidak perlu khawatir karena bahasa yang satu ini selain tidak berhubungan dengan reptil tersebut, juga sangat dapat diandalkan. Anda tidak akan menjadi buas walau berhubungan dekat dengan yang satu ini.

Python dikembangkan oleh **Guido van Rossum** (guido@python.org) pada awal tahun 1990 di Stichting Mathematisch Centrum (<http://www.cwi.nl>). Python sendiri dapat dikatakan merupakan penerus dari bahasa ABC. Pada tahun 1995, sebagian besar pengembangan Python dilakukan di Corporation for National Research Initiatives (<http://www.cnri.reston.va.us>). Di CNRI ini pula, beberapa versi Python dirilis. Pengembangan Python saat ini dinaungi oleh Python Software Foundation.

Dan bicara lisensi, bagi Anda yang peduli, terutama dari sisi bisnis, Python menawarkan keuntungan yang luar biasa. Beberapa rilis Python berada dalam bendera lisensi yang berbeda-beda. Akan tetapi, semua lisensi Python adalah *open source*. Hampir semua lisensi Python kompatibel dengan GPL, kecuali Python versi 1.6, 2.0, 1.6.1, dan 2.1. Lisensi Python mengizinkan Anda untuk mendistribusikan versi modifikasi tanpa mengharuskan Anda untuk membuka source hasil modifikasi tersebut. Python bisa Anda dapatkan di <http://www.python.org>.

Python sendiri adalah sebuah interpreter. Python dapat diandalkan untuk mengerjakan hal-hal yang sangat rumit dan berhubungan dengan perhitungan dan ketelitian tingkat tinggi. Akan tetapi, karena Python adalah interpreter, dalam beberapa kasus, Python harus didampingi oleh bahasa pemrograman lain. Salah satu teman dekat Python adalah Java. Bagi Anda yang hanya tertarik untuk setia dengan Python dalam segala kondisi, gunakan

Psyco untuk mempercepat eksekusi program yang dihasilkan.

Mari beranjak ke dukungan *platform*. Python dapat dijalankan di hampir semua platform, mulai Linux, Windows, sampai Mac. Mulai komputer biasa sampai PDA. Sekali menulis kode dan apabila gaya pemrograman Python yang baik diterapkan, program yang dihasilkan sudah dapat dijalankan langsung di platform lain.

Anda mulai tertarik dengan Python, dan Anda bertanya: Saya belum pernah belajar pemrograman. Saya juga tidak memiliki banyak waktu. Dapatkan saya mempelajari Python? Jawabnya adalah Ya. Instalasi dan silakan mencoba! Python termasuk bahasa yang mudah dipelajari. Walau sayangnya, bagi Anda yang sering menggunakan C/C++, sintaks Python akan terlihat cukup aneh.

Python sendiri termasuk ke dalam *general purpose programming language*. Hampir semua tugas pemrograman dapat diselesaikan dengan bahasa ini. Mulai dari pemrograman di lingkungan sistem operasi Linux sampai pemrograman berbasis web. Anda bahkan dapat membuat sebuah *web server* sederhana dengan beberapa baris kode.

Sayangnya, di luar semua kekuatan Python, ada hal lain yang harus dibayar: performa. Performa program yang ditulis Python jauh berada di bawah program yang ditulis dengan C/C++. Rasionya dapat mencapai 1:5 sampai 1:10. Guido van Rossum menuliskan beberapa artikel tentang perbandingan antara Python dan bahasa lainnya. Anda dapat menemukan

artikel-artikel tersebut di situs Python.

Kemudian kita akan beranjak ke dunia pemrograman berorientasi objek. Apabila Anda pernah menggunakan Java, Anda akan melihat sangat banyak persamaan antara konsep objek di Java dan di Python. Ya, Python sangat mendukung pemrograman berorientasi objek. Semuanya merupakan objek di Python. Walau demikian, berbeda dengan Java, Anda tidak perlu membuat sebuah *class* secara eksplisit dalam pembuatan suatu aplikasi sederhana. Para developer yang menyenangi *structured programming* pun akan tetap merasa betah menggunakan bahasa yang satu ini.

Beberapa waktu yang lalu, jika Anda memilih Python untuk membuat aplikasi berbasis GUI menggunakan modul-modul standar, Anda akan memerlukan waktu cukup lama dalam mendesain antarmuka aplikasi. Bagi yang datang dari dunia Delphi misalnya, Python akan tampak sangat rumit dipakai. Untunglah, saat ini, untuk Rapid Application Development, sebuah program dengan nama Boa Constructor telah lahir mendukung Anda untuk semakin mencintai Python. Dengan Boa, Anda seperti menemukan Delphi dengan dialek Python.

Masih soal pembuatan aplikasi GUI, Anda mungkin tidak menyukai Tkinter yang datang sebagai GUI toolkit *default* Python. Jangan khawatir, di luar sana, dukungan berbagai GUI toolkit untuk Python dapat ditemukan di mana-mana. Sebut saja pygtk dan pyqt untuk GTK+ dan QT.

Isu bahasa pemrograman: register vs stack

Java, Perl 5, dan Python termasuk *stack-based system*. Dalam suatu *stack-based system*, semua operasi mendaftarkan dirinya pada *stack*. Sebagai contoh, operasi penambahan dua bilangan. Letakkan kedua bilangan tersebut pada *stack* dan berikanlah perintah "tambah", yang akan mengambil dan menjumlahkan kedua bilangan tersebut dan meletakkan hasilnya kembali di *stack*. Kekurangannya adalah sistem harus menghabiskan banyak waktu bermain-main dengan *stack*.

Ada pula sistem lainnya, yaitu sistem *register*. Sistem yang satu ini tidak bermain-main dengan *stack*, kecuali kode Anda sendiri yang menginginkannya. Salah satu kekurangannya adalah kompleksitas pada penggunaan *register*. Contoh *register-based system* adalah Parrot. (Sumber: *Linux Magazine* April 2003)

Apabila Anda ingin tetap menggunakan bahasa *stack-based system* sementara menginginkan kelebihan dari *register-based system*, umumnya Anda dapat menggunakan versi *stackless*-nya, sebagai contoh adalah *stackless Python*.



« Guido van Rossum, inovator python

Sehubungan dengan sifatnya sebagai interpreter, Python dapat

digunakan oleh aplikasi-aplikasi lain. Sebagai contoh, Anda dapat menuliskan script irc ketika sedang berkomunikasi di irc menggunakan XChat. Anda juga dapat menulis Python-Fu di GIMP.

Berbagai aplikasi pun telah dikembangkan dengan Python. Salah satunya adalah *installer* RedHat Linux yang berbasis GUI.

Kita semua telah setuju bahwa dengan mempelajari Python, kita tidak akan menjadi buas seperti halnya sebuah reptil. Yah, mungkin cukup buas dalam menghasilkan aplikasi dengan kualitas luar biasa, buas dalam kecepatan penulisan program, dan buas dalam skalabilitas. Sebenarnya, kenapa harus Python? Kenapa tidak nama yang lebih elegan? Yang tidak membuat orang harus mengernyitkan dahi terlebih dahulu? Semua itu kembali pada saat Guido harus memberi nama bahasa yang diciptakannya tersebut.

Guido sangatlah menyenangi acara Monty Python's Flying Circus, sebuah komedi BBC pada tahun 70-an. Dan dari sanalah asal nama bahasa pemrograman tangguh ini. Pendek, unik dan misterius. Guido sendiri tidaklah keberatan apabila kita ingin mengasosiasikan Python dengan reptil, berat 16 ton ataupun tikus.

Sekarang, mungkin banyak yang akan mempertanyakan kenapa Python memiliki fitur yang cukup aneh dan tidak lazim seperti indentasi? Kenapa pula harus menulis kode program dengan gaya menulis karangan di masa SD yang penuh dengan aneka ragam aturannya? Kenapa tidak membiarkan saja *whitespace* dan gunakan *token-token* yang sedikit masuk akal?

Apabila pascal punya *begin* dan *end*, kenapa Python tidak menggunakan saja *start* dan *end*? Mari kembali ke masa-masa sebelum tahun 1989, di mana Guido, *the benevolent dictator of Python* masih bekerja di CWI, tempat di mana beliau banyak belajar tentang desain

bahasa. Pada awalnya, beliau menggunakan bahasa ABC dan segera menyadari bahwa bahasa tersebut tidak dapat dikembangkan lebih lanjut.

Guido juga memilih untuk tidak mengembangkan Modula 2 dan Module 3. Yang terakhir ini adalah sumber inspirasi untuk sintaks dan semantik eksepsi dan berbagai fitur lainnya. Ketika Python digunakan pada Amoeba, sebuah *distributed* OS yang digunakan oleh Guido, berbagai hasil yang memuaskan segera di dapat. Dan untuk mewujudkan dunia yang lebih baik, bahasa ini pun dirilis untuk umum.

Saatnya menjelekkkan bahasa scripting. Dan seharusnya, Python pun harus dijelekkkan. Pendapat miring tentang bahasa scripting adalah ekstensibilitas dan skalabilitas. Pada pembahasan Tcl, hal ini menjadi salah satu sebab bagi **Richard Stallman** untuk tidak menggunakan Tcl. Di dunia nyata, Python sendiri cukuplah *extensible* dan *scalable*. Buktinya adalah dukungan bahasa-bahasa dan pustaka-pustaka lain.

Bukti yang lebih kuat adalah penggunaan Python untuk menulis ZOOPE, sebuah *Application Server* yang (setidaknya menurut penulis) sungguh luar biasa dan hampir tiada duanya! Python digunakan pada berbagai situasi yang menuntut dinamisme tinggi, kemudahan pemakaian, kekuatan, dan fleksibilitas. Python dapat sangat disejajarkan dalam pemrosesan teks, seperti halnya bahasa Perl.

Oleh karena itulah, Python juga dapat digunakan pada *system administration*. Dan salah satu kekuatan Python yang tidak diragukan lagi adalah pemrosesan bilangan (Guido juga seorang ilmuwan matematika, di samping ilmuwan komputer). Kombinasi antara Numeric Python dan database Oracle akan menghasilkan analisis statistik yang dapat digunakan untuk menganalisis bisnis Anda.

Untuk mengetahui seberapa serius Python, mari sejenak melihat ke aplikasi-aplikasi yang telah di bangun dengannya. Pertama-tama kita melongok ke ZOOPE. Kemudian ada Grail (sebuah web browser) dan KnowBot Operating

Environment (*distributed environment* untuk kode mobile). Ada pula *engine* untuk *virtual reality* di University of Virginia. Dan masih banyak lagi.

Karena kode Python cukup repot untuk ditulis dengan editor biasa dalam soal indentasi, mari sejenak melihat beberapa editor yang cocok dipakai untuk menulis kode Python. Pertama-tama adalah Emacs dan Vim. Pilih salah satunya dan Anda akan mendapatkan lingkungan penulisan kode yang menyenangkan. Anda mungkin akan mempertimbangkan juga FTE dan Glimmer. Selengkapny tentang editor untuk python dapat dibaca di <http://>



www.bofh.asn.au/~richard/editors.html.

Pengembangan masa depan Python dapat dilihat pada PEP (Python Enhancement Proposals) di <http://www.python.org/peps/>. Pengembangan Python saat ini berada di bawah kontrol Python Software Foundation.

Untuk saat ini, Python sangat stabil untuk digunakan, dan terbukti telah digunakan sejak lama oleh Google, Inc. Di masa mendatang, sangat mungkin terjadi perubahan pada spesifikasi bahasa. Python 3.0 akan menerapkan beberapa perubahan yang signifikan. Apabila Anda telah menikmati salah satu versinya, umumnya langkah *upgrade* telah cukup aman untuk mengatasi *bug*. Anda tidak harus selalu meng-*upgrade* ke versi yang lebih tinggi. Walau, untuk saat ini, pengguna Python sangat dianjurkan untuk menggunakan versi 2.2.x.

Dan terakhir, berikut ini adalah kode untuk mencetak tulisan 'Hello World' di console:

```
#!/usr/bin/python
print 'Hello World'
```

Tcl



Hampir selama 14 tahun di University of California di Berkeley, pindah ke Sun Microsystems selama beberapa tahun dan kemudian berpindah pula ke Scriptics. Ya, beliau adalah **DR. John K. Ousterhout**, sang inovator Tcl. Perjalanan panjang beliau dalam meningkatkan daya guna Tcl boleh diacungkan jempol. Mulai dari keterbatasan tipe daya sampai pada anjuran **Richard M. Stallman** untuk tidak menggunakan Tcl.

Ada apa dengan Tcl? Bahasa pemrograman yang bertempat tinggal di <http://www.tcl.tk> ini memiliki sejarah yang cukup panjang dan bergejolak. Pada awalnya, Ousterhout sangat memerlukan suatu bahasa yang sangat mudah dipakai, sangat sederhana dan mampu menjadi semacam perekat di antara berbagai fungsi bahasa tersebut.

Setelah mencari ke sana ke mari, akhirnya beliau memutuskan untuk menulis sendiri bahasa yang dibutuhkan tersebut pada tahun 1988. Perlu dicatat bahwa pada saat tersebut, Perl baru saja lahir, sementara Python mungkin belum berbentuk.

Kemudian lahirlah Tcl. Minus dukungan *array* dan *linked list*. Hal tersebut jugalah yang membuat RMS (terlepas dari berbagai faktor lainnya) menyerukan kepada masyarakat umum untuk tidak menggunakan Tcl dalam pemrograman di lingkungan yang produktif. Menurut beliau, dalam postingnya di salah satu milis pada bulan September 1994, Tcl hanyalah sebuah *extension* dan tidak memiliki skalabilitas yang tinggi.

Walau demikian, pendapat dari RMS juga ditentang oleh berbagai lapisan *developer* yang ada. Wajar saja, di era-era tersebut, bahasa pemrograman yang sangat lengkap cukup susah untuk ditemui (mungkin disebabkan oleh faktor kebutuhan pasar).

Lupakan sejenak masa lalu dan kita kembali ke saat ini untuk melihat kenapa Tcl mampu bertahan lebih dari 10 tahun. Bahkan Tk, pasangan setianya sampai

digunakan sebagai default GUI toolkit pada Python. Secara fungsional, saat ini, Tcl bisa dikatakan memiliki kemiripan yang cukup sama dengan berbagai bahasa pemrograman *scripting* lainnya. Sebut saja Perl misalnya. Tool Control Language, begitulah kepanjangan dari Tcl adalah bahasa yang akan membuat pekerjaan yang rumit pada C menjadi jauh lebih mudah. Ingin tahu alasan-alasan lain kenapa Tcl dipakai? Berikut ini beberapa di antaranya:

→ Pengembangan yang cepat

Anda dapat mengembangkan aplikasi dalam tenggang waktu yang relatif cepat, tentu saja apabila dibandingkan dengan C/C++. Jika aplikasi Anda melibatkan penggunaan GUI dan penanganan *string*, maka waktu pengembangan Anda akan menjadi lebih efektif lagi.

→ Dukungan GUI

Ousterhout menulis Tk pada akhir tahun 1988 dan sampai sekarang Tk masih menjadi GUI toolkit yang sangat dapat diandalkan. Aplikasi GUI yang Anda buat mungkin tidak akan secerah apabila dibangun dengan GTK+ ataupun QT, tapi sudah sangat cukup memenuhi kebutuhan mendasar. Tk juga termasuk salah satu GUI toolkit yang memiliki predikat relatif mudah dipakai

→ Aplikasi cross-platform

Tcl memiliki slogan yang cukup berani: *"Write Once, Run Anywhere"*. Ya, Tcl dapat dijalankan di berbagai *platform* seperti UNIX, Mac, dan Windows. Toolkit Tk yang disediakan pun dapat berjalan di berbagai platform tersebut.

→ Dapat dikembangkan

Anda ingin menambah fungsi baru? Tcl, sejak masa-masa awalnya selalu menjanjikan bahwa penambahan fungsi bukanlah menjadi masalah. Akan tetapi, beberapa pendapat di milis juga menyatakan bahwa sekali aplikasi Tcl telah menjadi begitu kompleks, perubahan strukturalnya adalah sebuah mimpi buruk berkepanjangan. Terlepas dari benar atau salah, hal yang satu ini mungkin juga perlu dipertimbangkan.

→ Terintegrasi dan fleksibel

Anda dapat saja menggunakan Tcl sebagai bahasa utama, kemudian menambahkan GUI, kemudian menambahkan dukungan jaringan atau mengintegrasikan penggunaan Java di dalamnya.



↑ John K. Ousterhout, inovator Tcl

→ Siap untuk dukungan enterprise

Menurut klaim di situsnya, dengan rilis 8.1nya, Tcl menjadi bahasa pertama yang cocok untuk digunakan pada kelas server aplikasi dan beberapa penggunaan pada level *enterprise*. Untuk membuktikannya, Tcl membungkus kerja sama antara *i18n*, *thread* yang aman, portabilitas, dukungan GUI yang baik, dukungan Internet dan database. Umumnya fitur-fitur tersebut juga terdapat di bahasa pemrograman lainnya.

→ Testing

Tcl juga cocok digunakan dalam otomatisasi *hardware* dan testing pada proses perancangan *software*. Dengan Tcl, Anda dapat dengan mudah terhubung ke hardware percobaan, menjalankan fungsi-fungsi testing, mengevaluasi hasil tes dan melaporkan kesalahan.

→ Mudah untuk dipelajari

Tcl adalah bahasa yang sangat sederhana. Programmer yang berpengalaman dapat membuat aplikasi yang menarik dalam hitungan jam. Para pendatang baru pun dapat mempelajari Tcl dengan cepat.

Advokasi Tcl cukuplah hebat. Menurut redaksi, beberapa hal yang dicatat sebagai keunggulan dalam advokasi tersebut juga dimiliki oleh bahasa pemrograman lainnya. Silakan mencoba dan buktikan sendiri. Tertarik untuk melihat contoh program tcl? Berikut ini adalah program "Hello World":

```
#!/usr/bin/tclsh8.3
```

```
puts "Hello World"
```



KATEGORI GUI TOOLKIT

■ GTK+



Katakanlah Anda termasuk orang yang membenci *command line* (dan itu wajar), dan Anda adalah seorang developer, maka

GUI Toolkit yang satu ini mungkin akan menarik perhatian Anda. Ya, GTK+ adalah kumpulan *widget* yang efisien dengan *look and feel* dirancang seperti Motif. GTK+ sendiri adalah singkatan dari GIMP Toolkit. GTK+ pada awalnya dibuat oleh dibuat oleh **Peter Mattis**, **Spencer Kimball**, dan **Josh MacDonald**. Anda bisa mendapatkan GTK+ di www.gtk.org. GUI toolkit yang satu ini dilisensikan di bawah bendera LGPL (Lesser GPL).

Apabila GTK adalah GIMP Toolkit, lalu dari mana asal tanda + nya? Peter Mattis menerangkan bahwa tanda + tersebut diberikan untuk membedakan versi orisinal dan versi yang baru (pada waktu itu). Salah satu perbedaannya adalah versi baru tersebut menambahkan fitur *object oriented*. Saat ini, umumnya, istilah GTK merujuk kepada GTK+.

GTK+ menyediakan beberapa fitur yang menarik. Ambil contoh tombol yang notabene merupakan widget standar di aplikasi GUI. Umumnya sebuah tombol mengandung label ataupun pixmap (ikon). Dalam GTK+, tulisan pada tombol tersebut bukanlah terkandung sebagai bagian integral dari *widget button*.

Akan tetapi, tulisan tersebut adalah sebuah widget yang dikandung oleh widget button. Dengan demikian, Anda dapat menambahkan widget-widget lain ke dalam *button* tersebut. Cukup fleksibel bukan?

GTK+ tidak bekerja sendiri. Anda perlu memiliki pustaka-pustaka lain supaya bisa bekerja lebih cepat dengan GTK+, terutama kalau Anda menggunakan bahasa C sebagai standar dari GTK+ dan aplikasi yang dikembangkan cukup kompleks. Umumnya Anda akan memerlukan GDK

(GTK+ Drawing Kit) dan GLib (G Libray). Anda juga akan memerlukan ATK dan Pango. Ya, GTK+ cukup terkenal dengan *dependency hell*-nya. Tetapi sekali lagi, pustaka-pustaka tersebut diperlukan apabila aplikasi yang ingin dibangun cukup kompleks.

Bicara soal platform lainnya, GTK+ cukup dapat diandalkan. Sebagai contoh, GTK+ juga dapat digunakan di sistem operasi Microsoft Windows. Selain itu, GTK+ juga menyediakan *binding* ke bahasa-bahasa pemrograman lainnya. Sehingga jika Anda tidak ingin menggunakan bahasa C, Anda tetap dapat menggunakan bahasa favorit Anda dan mendapatkan hasil yang indah. Sebut saja GTK- (GTKmm) untuk C++ dan Pygtk untuk Python.

Performa dari GTK+ dapat dikatakan cukup baik. Selain itu, GTK+ juga memiliki kemampuan untuk dikustomisasi dengan penggunaan theme sehingga tampilan yang baru bisa didapatkan tanpa harus melalui proses kompilasi ulang.

Sayangnya, GTK+ cukup rumit untuk dipelajari oleh pemula. Hal ini lebih disebabkan karena penggunaan bahasa C sebagai bahasa *default*. Sebagai solusinya, tetap gunakan bahasa favorit Anda. Cari binding untuk GTK+. Kerugiannya, aplikasi Anda akan berjalan sedikit lambat. Kombinasi Python/GTK+ atau Perl/GTK+ dapat menjadi senjata yang cukup ampuh.

Apabila Anda ingin mengembangkan aplikasi dengan GTK+, sebuah versi free dari buku **Havoc Pennington** bertajuk *GTK+/GNOME Application Development* dapat Anda baca di <http://developer.gnome.org/doc/GGAD/> Perbaikan dan informasi lain buku tersebut bisa dibaca di <http://pobox.com/~hp/gnome-app-devel.html>.

Havoc sendiri tercatat sebagai *hacker* yang sangat aktif mengembangkan keindahan *user interface* di GNOME. Apabila Anda menggunakan RedHat 8,

keindahan GUI yang Anda nikmati adalah bagian dari kerja keras beliau di RedHat.

Saat ini, aplikasi yang dibangun dengan GTK+ sudah sangat banyak. Salah satu contoh suksesnya adalah desktop environment GNOME. Berminat menambah koleksi aplikasi yang dibangun dengan GTK+? Mulailah dari pembuatan *window* dan sebuah tombol dengan tulisan "Hello World" berikut ini:

```
#include <gtk/gtk.h>

int main(int argc, char *argv[] )
{
    GtkWidget *window;
    GtkWidget *button;

    gtk_init(&argc, &argv);

    window = gtk_window_new
(GTK_WINDOW_TOPLEVEL);
    gtk_container_set_border_width
(GTK_CONTAINER (window), 10);

    button = gtk_button_new_with_label
("Hello World");
    gtk_container_add (GTK_CONTAINER
(window), button);

    gtk_widget_show (button);
    gtk_widget_show (window);

    gtk_main ();

    return(0);
}
```

GUI Toolkit lain-lain

Sama seperti bahasa pemrograman, jumlah GUI toolkit di dunia open source pun lumayan banyak. Mulai yang berat seperti QT, GTK+ dan TK, sampai yang relatif ringan seperti minigui dan picogui. Pertanyaan klasiknya: pilih mana?

Hal tersebut, sama seperti pemilihan bahasa pemrograman, kembali ke tujuan masing-masing. Jika tujuan Anda adalah aplikasi yang *user friendly*, indah, luar biasa dan boleh berlambat-lambat ria, gunakan QT. Paling tidak GTK+. Kalau tujuan Anda adalah fungsi dan sedikit keindahan, TK cukup menarik.

■ QT



Beberapa waktu yang lalu majalah *InfoLinux*

mengadakan survai, yang mana salah satu item yang disurvei adalah *desktop* favorit. Dan tidak diragukan lagi, pilihan jatuh pada KDE. Dan bicara soal KDE, kita tidak mungkin tidak membicarakan pula QT. QT lah yang membangun KDE. Andil QT sangatlah besar untuk menciptakan desktop Linux yang semakin ramah dari hari ke hari. QT dibuat oleh perusahaan software Norwegia, TrollTech, dan dilisensikan di bawah bendera GPL/QPL untuk komunitas *open source*.

Bagi Anda yang ingin turut memberikan kontribusi dalam pengembangan *software-software* ramah dan *user friendly*, Anda mungkin perlu mempertimbangkan untuk menggunakan QT. Berikut ini adalah beberapa alasan di antaranya.

→ Kemampuan multiplatform

Bila target aplikasi Anda adalah Microsoft Windows, Linux, Unix, Mac OS, dan embedded Linux, maka gunakan QT. Kode yang Anda tulis dapat dijalankan pada platform-platform tersebut tanpa harus mengalami berbagai masalah yang menjengkelkan. Pada Mac OS yang telah terbukti menjadi juara di desktop pun, Qt terbukti semakin akrab. Dan Apabila Anda memiliki PDA Sharp Zaurus, QTopia telah bertengger dengan manis di sana, siap membantu mengelola berbagai kegiatan Anda. TrollTech dalam *press release*-nya menjanjikan bahwa kode yang Anda tulis dengan QT akan sangat fleksibel dan dapat dijalankan di berbagai platform. Dan untuk itu, Anda hanya perlu mempelajari satu API. Satu!

→ Integrasi dengan Motif

Modul QMotif pada QT 3.1 akan memanjakan *developer* yang akan memasukkan kode QT mereka ke dalam aplikasi Motif langkah demi langkah. Tidak akan ada pengembangan ulang yang akan memakan waktu lama. Sekali

suatu aplikasi telah di migrasi ke QT, maka Anda akan mendapatkan kemampuan untuk *multiplatform* dan fleksibilitas yang tinggi.

→ Integrasi dengan ActiveX

Module ActiveX mampu untuk menjalankan kontrol ActiveX pada aplikasi QT. Anda bahkan dapat membangun kontrol ActiveX dengan QT. Dan semua hal tersebut dilakukan dengan mudah dan fleksibel.

→ Maintenance yang tidak rumit

Dengan menghapuskan *source tree* yang banyak, QT menghapuskan biaya pemeliharaan yang bisa menjadi sangat banyak. Dengan hanya satu *tree source* saja, Anda hanya perlu memelihara satu *tree*, mengeluarkan biaya pemeliharaan yang lebih sedikit, dan waktu yang lebih banyak untuk hal lainnya.

→ Tampilan Native setiap platform

QT tidak memerlukan *virtual machine* ataupun lapisan emulasi lainnya. QT memiliki akses *low level* ke fungsi-fungsi grafik seperti yang dilakukan oleh aplikasi *native* lainnya. Dengan demikian, aplikasi akan berjalan lebih cepat. Selain itu, Anda akan mendapatkan pula *look and feel native* setiap *platform* sehingga keindahan akan tetap terjaga. Aplikasi QT yang berjalan di Mac OS 10.2 misalnya, akan memiliki keindahan seperti aplikasi nativenya. Dan jika Anda tidak menyukai tampilan tersebut, hias saja aplikasi Anda dengan penggunaan *theme*.

Bagi Anda yang terbiasa bermain dengan Delphi ataupun Visual Basic, Anda tidak harus mengetikkan semua kode Anda tanpa bantuan RAD yang baik. Lupakan pengetikkan kode program vim. Dengan QT Designer, RAD yang disertakan bersama QT, Anda dapat merancang antarmuka aplikasi Anda dalam beberapa klik dan *drag mouse*.

Bahasa yang digunakan oleh QT adalah C++. Apabila Anda tidak

menyukai bahasa tersebut dan lebih menyukai bahasa-bahasa *scripting* lainnya, binding untuk bahasa-bahasa lain cukup banyak. Apabila GTK+ ada *pygtk*, maka QT memiliki *pyqt* untuk penggunaan bahasa Python. Pokoknya, gunakan saja bahasa Anda dan serahkan kepada QT untuk penanganan antarmuka yang ramah.

Beberapa pendapat miring di Internet mempermasalahkan soal masa depan GUI toolkit yang satu ini sehubungan dengan kelangsungan hidup Trolltech. Untuk yang satu ini, Anda tidak perlu khawatir. Andaikata Trolltech bangkrut pun, QT akan jatuh ke dalam lisensi semacam BSD, menurut **George Staikos**, salah seorang hacker KDE. Dan proyek KDE pun akan tetap berlanjut.

Lebih lanjut bagi Anda yang bergerak di dunia bisnis, lisensi adalah hal yang cukup harus diperhatikan. Apabila *software* yang Anda tulis akan dilepas secara *non-free/proprietary*, maka Anda harus terlebih dahulu membeli QT versi Professional ataupun *enterprise*.

Dan Anda harus membeli QT tersebut sejak awal ketika Anda akan mengembangkan aplikasi non-free. Anda tidak dapat menggunakan versi Free dari QT pada saat mengembangkan aplikasi dan baru membeli versi profesionalnya ketika aplikasi siap Anda lepas. Di luar itu, semua aplikasi yang dikembangkan dengan versi free dari QT harus dilepas sebagai free software. Rasanya cukup *fair* bukan?

Berikut ini adalah contoh *source code* untuk membuat *window* dan sebuah tombol dengan tulisan "*Hello world*":

```
#include <qapplication.h>
#include <qpushbutton.h>

int main( int argc, char **argv )
{
    QApplication a( argc, argv );
    QPushButton hello( "Hello world!", 0 );
    hello.resize( 100, 30 );
    a.setMainWidget( &hello );
    hello.show();
    return a.exec();
}
```