

Pengembangan Aplikasi di Linux

Noprianto

Linux adalah surga bagi para programmer. Tersedia banyak bahasa pemrograman, pustaka dan tool-tool berkualitas yang siap digunakan.



Dalam penggunaan komputer, sistem operasi saja tidak cukup untuk membantu memenuhi kebutuhan kita. Berbagai aplikasi umum diperlukan. Sebagai contoh, kita membutuhkan paket office untuk membantu pekerjaan kantor seperti mengetik, membuat laporan, mengisi data, membuat presentasi dan lain sebagainya. Kita membutuhkan web browser agar kita dapat berselancar di internet. Berbagai paket manajemen email juga dibutuhkan agar kita bisa mengirim dan menerima email. Aplikasi multimedia dibutuhkan agar kita bisa menikmati multimedia dengan bantuan komputer. Banyak sekali aplikasi yang tersedia dan dapat digunakan. Beberapa diantaranya benar-benar gratis, beberapa diantaranya membuat kita harus merogoh kocek.

Terkadang, aplikasi-aplikasi umum belumlah cukup. Sebuah pabrik misalnya. Pabrik tersebut tidak hanya membutuhkan aplikasi office atau web browser semata. Mereka membutuhkan aplikasi khusus yang mengerti kebutuhan mereka dan dapat membantu mereka agar dapat lebih produktif. Kita mengenal ERP yang membantu perusahaan dari ujung sampai ujung lainnya. Dengan menggunakan ERP

tersebut, sebuah pabrik dapat bekerja lebih efektif dan efisien. Walaupun, terkadang harga sebuah ERP bisa membuat kita pusing.

Terdapat banyak aplikasi umum di Linux. Namun, software-software dengan kualitas lebih baik selalu dibutuhkan. Oleh karena itu, mungkin sekali terdapat puluhan software untuk fungsi yang sama. Sementara, aplikasi khusus tidak pernah menjadi terlalu banyak. Banyak perusahaan yang menginginkan solusi teknologi informasi yang lebih personal dan tidak hanya mengandalkan software-software umum. Umumnya, hal ini disebabkan karena kebutuhan.


Apabila Anda tertarik untuk mengembangkan aplikasi di Linux, baik aplikasi umum ataupun aplikasi khusus, berbagai hal perlu dipertimbangkan agar dapat mengembangkan aplikasi secara lebih cepat, lebih baik, dan pada akhirnya, dapat membantu memenuhi kebutuhan.

Ulasan utama kali ini mencoba untuk mengangkat hal-hal seputar pembuatan aplikasi di Linux. Pembahasan difokuskan pada beberapa hal penting yang menjadi kunci pembuatan aplikasi. Pertama-tama, kita akan melihat soal pemilihan bahasa

pemrograman. Di dunia free software, tersedia begitu banyak bahasa pemrograman. Setelah itu, kita juga membahas detail tentang pembangunan user interface. Pembahasan meliputi CLI, TUI, GUI dan aplikasi web. Pembahasan mengenai database juga tidak terlupakan. Berbagai database termasuk RDBMS dan OODB juga dibahas. Tak lupa, database embedded dimasukkan sebagai salah satu topik spesial dalam penggunaan database.

Hal-hal seperti lisensi juga merupakan masalah penting. Sama seperti halnya bagaimana membuat aplikasi yang dapat memiliki pasar yang luas. Penggunaan teknologi standar, i18n, dan kemungkinan cross-platform adalah hal-hal yang penting agar aplikasi kita dapat digunakan oleh berbagai pihak.

Untuk pembangunan aplikasi yang lebih cepat, ada baiknya kita menggunakan berbagai tool yang memang telah tersedia. Sebagai contoh, debugger, document generator, IDE, diagram modeller dan version control.

Pada akhirnya, tak lupa isu distribusi juga akan kita bahas. Bagaimana cara kita men-deliver aplikasi kita adalah hal yang tak kalah penting. Selamat membaca! 

Pemilihan Bahasa Pemrograman

Untuk melakukan pemrograman, kita akan memiliki bahasa pemrograman terlebih dahulu. Di dunia free software, terdapat banyak sekali bahasa pemrograman. Anda bisa memilih yang paling sesuai dengan kebutuhan. Beberapa contoh bahasa pemrograman yang populer diantaranya C/C++, Java, Pascal, Perl, PHP, Python, TCL, dan lain sebagainya.

Bukan masalah besar

Di dunia Windows, pemilihan bahasa pemrograman menjadi masalah yang cukup besar. Tak jarang, untuk pemrograman berbasis GUI, kita mendengar seteru antara pemakaian Delphi atau Visual Basic. Memilih Delphi berarti akan menggunakan gaya dan segala hal yang berhubungan dengan Delphi. Begitu juga dengan penggunaan visual Basic. Sangat susah untuk menggunakan komponen native Visual Basic di Delphi. Atau sebaliknya. Oleh karena itu, di dunia Windows, pemilihan bahasa pemrograman adalah hal yang penting. Terutama di pemrograman berbasis GUI, karena komponen pembangun GUI merupakan bagian integral dari bahasa tersebut.

Agak sedikit berbeda di Linux dan free software. Jarang sekali terdapat implementasi suatu bahasa pemrograman yang datang satu paket dengan GUI toolkit sendiri. Kita melihat bahwa implementasi C/C++ oleh GCC sama sekali tidak datang bersama pustaka untuk membangun GUI. Pascal yang diimplementasikan oleh Free Pascal juga sama. Perl juga sama. PHP juga sama. Python menggunakan pustaka Tk, namun bukan dikembangkan secara khusus untuk Python.

Kalaupun datang dengan GUI toolkit sendiri, umumnya, toolkit tersebut juga bisa diakses oleh bahasa pemrograman lain. Sebagai contoh, C++ yang diimplementasikan oleh QT adalah sebuah pustaka sekaligus bahasa lengkap, termasuk untuk membangun aplikasi berbasis GUI. Namun, pustaka QT juga bisa diakses oleh Python misalnya. Contoh lain adalah Tk, walaupun

diasosiasikan dengan TCL, pustaka tersebut dapat digunakan oleh Python misalnya.

Apabila memang ingin membangun aplikasi berbasis GUI, banyak pustaka yang didedikasikan khusus untuk itu, dan umumnya, binding untuk berbagai bahasa juga telah tersedia.

Dengan kata lain, apabila hanya masalah GUI, pemilihan bahasa pemrograman bukanlah masalah penting. Alasan lain seperti ketersediaan pustaka yang banyak juga bukan alasan yang terlalu penting.

Sebagai contoh, di PHP, mudah sekali untuk melakukan konektivitas database. Hampir setiap database besar dan terkenal didukung oleh PHP. Python mungkin agak susah untuk melakukan hal yang sama. Perl mungkin juga agak susah untuk melakukan hal yang sama. Namun, banyak implementasi dari pihak lain yang mengembangkannya untuk banyak bahasa pemrograman. Sehingga, walaupun dalam distribusi resminya Python tidak menyertakan dukungan untuk suatu database misalnya, kita bisa mendapatkannya dari pihak ketiga.

Pustaka-pustaka lain yang tidak datang bersama suatu bahasa pemrograman umumnya diimplementasikan oleh pihak ketiga dan dapat digunakan secara bebas.

Dalam kasus yang parah, sampai-sampai tidak ada pustaka dari pihak ketiga pun, kita masih bisa melakukan dynamic loading suatu pustaka sistem untuk mendapatkan suatu fungsi. Dan umumnya, dynamic loading pustaka sistem didukung oleh berbagai bahasa pemrograman.

Memilih sesuai kebutuhan

Pemilihan bahasa pemrograman di Linux lebih disesuaikan kepada kebutuhan. Tidak ada bahasa yang paling baik diantara semua bahasa pemrograman. Untuk mencari yang paling baik, bisa-bisa pemrograman tidak pernah dilakukan karena pencarian tersebut seperti pencarian yang tak kunjung selesai.

Apabila Anda membutuhkan kecepatan tinggi, kedekatan dengan sistem dan rela


untuk sedikit bersusah payah, maka C/C++ adalah pilihan yang baik. Keunggulan memilih bahasa ini sangatlah banyak. Berbagai pustaka sistem dapat Anda gunakan. Dengan demikian, Anda tidak perlu berpusing-pusing untuk memikirkan dukungan pustaka tertentu. Pengguna Java atau Python misalnya, mungkin akan kebingungan apabila tidak terdapatnya binding untuk suatu pustaka tertentu. Bagi Anda, hal tersebut bukanlah masalah. Cukup gunakan saja header-header yang diperlukan untuk mencapai hasil yang diinginkan.

Sementara, jika fleksibilitas, kemudahan serta tersedia banyak cara untuk melakukan sesuatu, Anda bisa menggunakan Perl. Perl sangatlah fleksibel, relatif mudah digunakan serta dapat melakukan suatu tugas dengan berbagai cara. Perl juga identik dengan administrator sistem.

Python dapat Anda pilih karena ketersediaan banyak pustaka dan binding, tipe data yang kaya, mudah dipelajari dan kode program yang bersih. Python juga cocok untuk digunakan pada aplikasi skala besar.

Java apalagi. Java adalah bahasa yang sangat bagus. Tapi, sebaiknya Anda memiliki konsep Object Oriented Programming yang kuat. Sediakan waktu lebih untuk belajar. Java adalah teknologi yang maju dan matang. Selain itu, juga cocok digunakan untuk aplikasi skala besar ataupun kecil.

TCL dan Pascal terkenal karena mudahnya. Pengguna Delphi akan cocok dengan Pascal. Pengguna Python mungkin akan cukup cocok dengan TCL. Sementara, PHP mungkin boleh dikatakan sebagai bahasa pemrograman web paling populer (di dunia free software).

Lihatlah area kebutuhan Anda. Baca-baca referensi, yakinkan mana yang paling cocok dengan gaya Anda, dan pilih satu. Pelajari dan selesaikan kebutuhan Anda. Semuanya bagus. Asalkan benar-benar dialami. Hanya, yang satu lebih bagus di satu sisi, sementara yang lain lebih bagus di sisi lain. 

Pembangunan user interface

Program yang dihasilkan tentu tidak hanya selalu digunakan oleh pembuatnya. Yang menggunakan umumnya adalah pihak lain. Untuk itu, user interface yang masuk akal perlu kita kembangkan agar program kita dapat digunakan oleh berbagai pihak. Untuk user interface, kita mengenal beberapa kategori, mulai dari command line interface sampai Graphical user interface.

Commad Line Interface

Banyak sekali program di dunia free software yang dibangun dengan commad line interface. Berbagai utilitas sistem boleh dikatakan datang dengan interface ini. Umumnya, satu program memiliki lebih dari satu opsi dan dijalankan dengan cara seperti ini:

```
ls -al
ls --all
ls -help
```

Apabila aplikasi yang Anda bangun memang tidak membutuhkan kemudahan pemakaian atau keindahan, maka command line interface adalah pilihan yang paling cocok. Target pengguna program Anda adalah pengguna yang bisa membaca manual program. Atau, paling tidak, dapat membaca fasilitas help program. Semua itu artinya, Anda perlu menyediakan manual atau paling tidak fasilitas help yang cukup memadai.

Sebagai seorang programmer, Anda tidak melakukan parsing command line secara manual. Tidak perlu membaca parameter yang dimasukkan kemudian melakukan pemisahan dan pemeriksaan secara manual. Umumnya, command line interface dikembangkan dengan pustaka getopt. Dengan menggunakan pustaka tersebut, Anda hanya perlu mendefinisikan berbagai opsi yang valid dan aksi yang berhubungan dengannya. Pemeriksaan dan parsing akan dilakukan sendiri. Binding untuk pustaka getopt tersedia untuk hampir semua bahasa pemrograman di dunia free software. Bagi Anda yang melakukan pemrograman dengan shell script sekalipun,

Anda masih bisa menikmati fasilitas ini. Hal ini disebabkan karena terdapat program getopt dan getoptsp yang berguna untuk melakukan parsing argumen dan opsi program.

Jadi, walaupun membuat aplikasi berbasis command line, kita tetap perlu memperhatikan pengguna program kita. Jangan pernah membuat program berbasis command line namun tidak menyertakan help ataupun manual.

Text User Interface

Berbeda dengan command line interface yang membiarkan penggunanya menggunakan opsi-opsi program untuk melakukan tugas tertentu, maka text user interface akan menampilkan berbagai window dan menu menarik (dalam modus text) untuk penggunanya. Umumnya, aplikasi dengan text user interface juga memiliki fasilitas untuk pemberian opsi. Namun, secara default, apa yang ditampilkan adalah interface yang cukup menarik. Contoh yang paling baik adalah Midnight Commander.

Di dunia DOS atau Windows, kita mengenal berbagai ungkapan bahwa pembuatan aplikasi dengan text user interface sudah tidak zaman lagi karena semuanya akan berbasiskan GUI. Kondisi ini sedikit berbeda di dunia free software. Tidak semuanya harus GUI. Terkadang, untuk kalangan tertentu, TUI lebih disukai. Ringan, cukup bagus dan kalau mau, mouse bisa digunakan.

Tentunya, tersedia banyak pustaka untuk membangun TUI ini. Sebut saja ncurses dan newt. Keduanya akan membantu Anda untuk membangun TUI dengan mudah. Implementasi untuk berbagai bahasa pemrograman juga tersedia.

Graphical User Interface

Ini adalah permasalahan yang cukup sering ditemui di dalam dunia Linux. Di dunia Windows, dengan Delphi atau Visual Basic misalnya, pembuatan GUI benar-benar bukan masalah. Desain user interface

bukan masalah sama sekali, bahkan terkadang tidak perlu coding sama sekali.


Di dunia Linux, apa yang mampu dilakukan oleh Delphi atau Visual Basic jelas masih jauh untuk dicapai. Namun, hal ini lebih disebabkan separasi antara bahasa pemrograman dan GUI toolkit. Dengan adanya separasi ini, pengguna berbagai bahasa pemrograman dapat membangun GUI dengan berbagai GUI toolkit.

Berbagai GUI toolkit populer yang pantas dipertimbangkan adalah GTK+, QT, Tk, wxWindows (yang akan berubah namanya menjadi wxWidgets). Dan, banyak sekali bahasa pemrograman yang mendukung toolkit-toolkit tersebut.

Contoh penggunaan GTK+ yang baik adalah desktop dan aplikasi-aplikasi GNOME. Untuk QT, contoh terbaik adalah KDE dan aplikasi-aplikasinya. Aplikasi yang dibangun dengan Tk juga cukup banyak. TK juga terkenal sebagai toolkit yang lengkap dan mudah digunakan. Selain itu, terdapat berbagai ekstensi Tk yang menambah kelengkapan widgetnya. WxWindows sendiri adalah pustaka populer untuk aplikasi GUI cross platform yang menawarkan tampilan native.

Pendekatan RAD seperti Delphi dan Visual Basic yang paling bagus barangkali adalah QT Designer. QT sendiri juga datang dengan versi enterprise yang mampu melakukan apa yang dilakukan oleh Delphi atau Visual Basic. Contoh lain barangkali Boa constructor untuk Python, menggunakan toolkit wxWindows. Khusus untuk Java, implementasinya cenderung lengkap dan memiliki GUI toolkit sendiri.

Web application

Untuk membuat aplikasi web, tentunya, kita dapat mengetikkan kode HTML sendiri. Namun, pertimbangkan juga untuk menggunakan berbagai tool-tool untuk mendesain user interface sehingga Anda tidak perlu membuat semuanya sendiri. Anda hanya mengisikan fungsi yang diperlukan saja. Berbagai aplikasi yang dapat merancang user interface dengan cukup bagus adalah Mozilla Composer dan OpenOffice.org. 

Penggunaan Database

Database selalu diperlukan apabila Anda membangun aplikasi bisnis atau aplikasi yang melibatkan penggunaan dan pengolahan data. Berbagai database siap pakai tersedia di dunia free software. Tinggal pilih yang mana, karena kebanyakan bahasa pemrograman tahu cara untuk berkomunikasi.

RDBMS

Jenis yang satu ini, relational database management system, adalah database yang cukup umum dan tergolong cukup senior. Implementasinya banyak. MySQL, MaxDB, PostgreSQL dan Oracle adalah contoh terbaik. Ketiganya cocok untuk digunakan dan mampu melayani data jumlah besar.

Apabila Anda menggunakan PHP, maka MySQL dan PostgreSQL bukanlah masalah. Sejak awal-awal PHP mulai berkibar, dukungan untuk kedua database canggih ini telah kita dapatkan. Bahkan, PHP sempat identik dengan MySQL. Hampir semua bahasa mampu berbicara dengan database ini. C/C++, Java, Perl, Python, TCL mampu melakukannya. Untuk bahasa yang tidak mendukung secara langsung, MySQL menyediakan sebuah pustaka client yang dapat digunakan untuk berbicara dengan MySQL Server. Jadi, selama suatu bahasa dapat melakukan dynamic loading, harusnya tidak akan ada masalah.

OODB

Yang satu ini baru-baru terkenal, walaupun implementasinya telah dilakukan beberapa waktu yang lalu. Ya, object oriented database saat ini semakin populer. Namun, OODB bukan dibuat untuk menggantikan RDBMS. Banyak sekali kelebihan RDBMS yang tidak dimiliki atau susah diimplementasikan di OODB. Contoh OODB yang bagus adalah ZODB, Zope Object Database, yang digunakan oleh Zope, sebuah application server berbasis Python.

Tentunya, pengguna Python dapat dengan mudah menggunakan database yang satu ini. Penggunaan oleh bahasa-bahasa lain seharusnya juga bisa dilakukan dengan tidak rumit.

Special: Embedded database engine

Masih ingatkah Anda akan Foxpro, clipper dan bahasa sejenis yang menawarkan kemudahan dan kesederhanaan? Satu aplikasi bisa memiliki beberapa tabel dimana setiap tabel disimpan di dalam setiap file. Untuk mengakses file tersebut, bahasa pemrograman telah mendukungnya sehingga tidak diperlukan layer tambahan. Dan yang penting, standalone. Cukup aplikasi itu sendiri saja yang diinstall. Tidak perlu melakukan koneksi pada server database. Apabila aplikasi ingin dipindahkan, maka copy saja semua file termasuk databasenya, dan jalankan tanpa masalah dimesin lain.

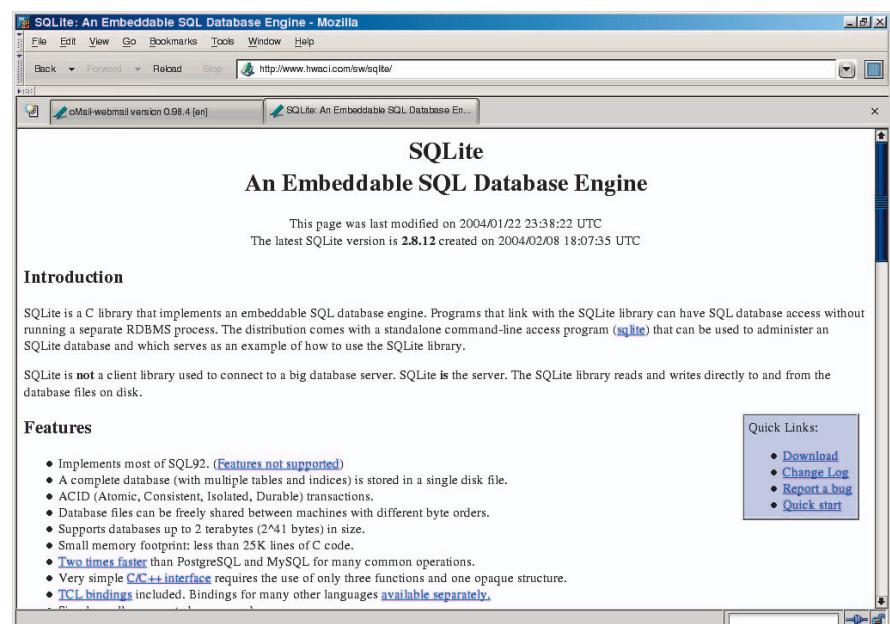
Bayangkan dengan MySQL dan PostgreSQL. Coba saja membuat aplikasi dengan kedua database server tersebut. Pembuatan memang terasa mudah. Tapi deployment dan backup akan terasa cukup susah, atau repot lebih tepatnya. Kita tidak dapat begitu saja mengopikan file-file database kita ke mesin lain begitu saja. Proses dumping ke file text misalnya, perlu dilakukan. Beberapa diantara mereka memang memiliki tool backup sendiri, namun proses deployment tetap saja lebih rumit.

Bukannya ingin mengatakan bahwa MySQL dan PostgreSQL tidak seru untuk digunakan. Mereka sangat seru untuk digunakan, jika aplikasi Anda benar-benar membutuhkan kemampuan database server tersebut.

Bagaimana jika apa yang Anda butuhkan hanyalah buku alamat atau organizer sederhana? Jelas kemampuan MySQL dan PostgreSQL belum diperlukan. Anda juga ingin agar aplikasi dapat dipindahkan ke mesin lain dengan mudah. Untuk menggunakan text file, jelas tidak dianjurkan sama sekali.

Untuk kasus tersebut, penggunaan embedded database sangat dianjurkan. Dengan demikian, Anda bisa membuat aplikasi dengan database serupa konsep foxpro, clipper dan lain sebagainya. Contoh database embedded yang sangat terkenal adalah SQLite dan BSddb.

SQLite misalnya, walaupun kecil, tetapi mampu bekerja dengan database seukuran 2 TB. Filenya pun hanya satu file untuk satu database. Perintah SQL pun didukung, walaupun cukup banyak fungsi yang tidak didukung seperti referential integrity. Kecepatannya sekitar 2 kali lebih cepat dari MySQL dan PostgreSQL untuk berbagai operasi.



▲ SQLite.

Lisensi

Lisensi tentu bukan permasalahan yang penting apabila Anda ingin membuat free software dan tidak ingin repot-repot soal legal. Cukup lisensikan saja di bawah lisensi GPL versi 2. Dan semuanya selesai. Yang penting adalah terjaminnya semua pihak untuk dapat menggunakan, mengembangkan, dan menyebarkan program yang Anda tulis. Masalah muncul ketika Anda membuat software proprietary, atau software dimana penggunaan dan pendistribusian memiliki aturan tertentu. Anda dapat menyesuaikan dengan berbagai lisensi yang tersedia, atau membuat lisensi sendiri.

Memilih lisensi

Kita mengenal bermacam-macam lisensi. Mulai dari lisensi proprietary sampai lisensi free software. Pemilihan lisensi menjadi penting apabila Anda ingin mengatur atau membatasi beberapa hal seperti penggunaan dan pendistribusian.

Sebagai contoh, Anda ingin agar aplikasi boleh disebarluaskan beserta source codenya, hanya modifikasi dan pendistribusian hasil modifikasi tidak diijinkan. Dengan demikian, source code diberikan hanya untuk dipelajari. Pada contoh di sini, segala source code dikontrol oleh Anda. Memilih lisensi GPL jelas tidak mungkin. GPL mengharuskan sebuah software dapat digunakan oleh siapa saja, dipelajari oleh siapa saja, bisa dimodifikasi oleh siapa saja sesuai keperluan, dan program Anda dapat didistribusikan oleh berbagai pihak. Sementara, hal tersebut jelas tidak memenuhi kebutuhan Anda. Beberapa solusi dapat diterapkan. Yang pertama adalah membuat lisensi sendiri (hal ini bukan langkah trivial, Anda perlu mengerti hal di luar pemrograman, atau lebih baik menggunakan jasa pihak lain yang khusus di bidang ini). Cara kedua lebih sederhana. Pelajari lisensi program lain, dan turunkanlah sesuai dengan keadaan Anda.

Bagi programmer free software yang tidak mementingkan lisensi, Anda bisa mempercayakan saja kepada lisensi GPL. Berbagai lisensi di kenal. Beberapa lisensi

berikut ini dianggap sebagai lisensi free software: GPL, LGPL, BSD, dan Artistic. Lisensi GPL misalnya, dapat dibaca secara online di <http://www.fsf.org/licenses/gpl.html>. Translasi ke beberapa bahasa bahkan telah tersedia. Sebagai contoh, translasi ke Bahasa Indonesia dapat Anda baca di <http://vlsm.org/etc/gpl-unofficial.id.html>.

Membuat free software

Free software memfokuskan kata free pada kebebasan, bukan pada harga nol. Arti kebebasan itu sendiri memiliki arti yang luas. Seseorang bisa menggunakan, mempelajari, memperbaiki dan mendistribusikan program yang Anda buat.

Lisensi yang terkenal sangat mengikat dalam free software adalah GPL. Dengan penggunaan lisensi GPL, semuanya harus bebas.

LGPL lebih sedikit longgar. Hal ini sangat berguna terutama ketika Anda membuat pustaka program. Beberapa pustaka yang dikembangkan lebih lanjut, untuk berbagai alasan misalnya, perlu tidak dilisensikan di bawah GPL.

BSD juga cukup ramah seperti halnya LGPL. Dengan BSD, Anda tidak perlu terikat erat seperti halnya GPL.

Membuat software proprietary

Umumnya, pembuat software proprietary memiliki aturan sendiri dalam penggunaan dan pendistribusian. Apabila software yang ditulis dapat digunakan oleh berbagai kalangan, maka lisensi dibuat lebih luas dan mengikat setiap kalangan penggunanya. Apabila hanya dapat digunakan saja, maka umumnya, pembuat software menerapkan semacam end user license agreement. Di Windows dan program-program proprietary yang berjalan di atasnya, kita dapat menemukan berbagai end user license agreement.

Apabila Anda membuat software proprietary, Anda bisa mempelajari lisensinya. Atau, beberapa program yang berjalan di Linux juga merupakan software

proprietary. Lisensi tersebut bisa Anda kembangkan sesuai kebutuhan.

Lisensi tidak harus panjang dan rumit serta penuh dengan istilah hukum. Untuk membuat lisensi, seseorang sebaiknya mengerti benar masalah hak kekayaan intelektual. Selain itu, beberapa proyeksi ke depan juga diperlukan agar lisensinya cukup luwes. Karena masalah lisensi selalu berhubungan dengan hukum, maka pastikanlah agar lisensi tersebut tidak menjadi senjata makan tuan.

Sekarang, bagaimana kalau Anda adalah seorang programmer yang bekerja pada suatu proyek yang Anda kerjakan sendiri dan ingin melepas proyek tersebut sebagai software yang source codenya tertutup namun binarynya dapat digunakan secara meluas? Dalam kasus ini, seharusnya lisensi Anda tidak akan terlalu rumit. Pastikan saja bahwa segala usaha dekompile adalah ilegal. Jangan lupa untuk mengatur masalah redistribusi di mana seorang penerima software mendistribusikan kembali software Anda. Bahwa pengguna software Anda tersebut menjualnya atau hanya mengopikannya kepada rekannya yang lain, tentunya harus juga untuk diperhatikan.

Bagi Anda yang ingin membuat program-program yang benar-benar mengharuskan pengguna untuk membayar, beberapa model dapat Anda kontek. Ada lisensi yang benar-benar mengharuskan nilai per mesin. Ada pula yang menitikberatkan pada nilai per pengguna. Dan pengguna yang bersangkutan dapat melakukan instalasi pada beberapa mesin.

Bermacam-macam lisensi proprietary diterapkan. Tentunya, semua sah-sah saja. Namun, pembuat software harus cerdas. Perhatikan juga lisensi dengan target pasarnya. Terkadang, memberikan lisensi yang lebih longgar akan berakibat lebih baik. Pokoknya, jangan sampai Anda pusing-pusing membayar suatu pihak untuk mengurus masalah lisensi super ketat sementara software Anda tersebar di mana-mana. Bagus untuk promosi dan dominasi produk memang, namun, lisensi Anda menjadi kurang penting. ☹

Tools pendukung

Apabila Anda adalah seorang programmer, tentu saja Anda bisa membuat sendiri debugger, document generator, editor, diagram modeller dan lain sebagainya. Tapi, akan lebih baik memfokuskan diri kepada fungsi utama program. Selebihnya, gunakan berbagai tool yang tersedia luas untuk mendukung program kita.

Di dunia pemrograman Windows, sebuah produk studio pemrograman umumnya datang dengan segala tool yang dibutuhkan untuk mempermudah pembuatan program. Sebagai contoh, selain datang dengan implementasi bahasa pemrograman, datang pula debugger, IDE, version control dan lain sebagainya. Kelengkapan produk akan menentukan seberapa banyak yang akan membeli produk tersebut.

Di Linux berbeda cukup jauh. Umumnya, implementasi bahasa pemrograman hanya akan memfokuskan dirinya pada bahasa pemrograman dan berbagai fitur kompilasi pada umumnya. Sementara, pengguna mengetikkan kode programnya dengan editor lain, mendebugnya dengan GDB, menggenerasi dokumen dengan document generator lain, membuat diagram dengan umbrillo, dan menggunakan CVS sebagai version control.

Bahkan, hebatnya, banyak user yang menggunakan suatu bahasa pemrograman, dan mendesain graphical user interface dengan program lain. Sebagai contoh adalah Python dan Boa-constructor.

Berikut ini, kita akan membahas beberapa tool yang bisa Anda gunakan untuk mempercepat pembuatan dan meningkatkan kualitas program yang Anda buat.

Building

Berbagai tools untuk pembangunan executable dapat Anda gunakan untuk memudahkan proses ini. Terutama, apabila Anda adalah programmer tunggal yang harus mengurus logika program, dokumentasi dan lainnya.

- **autoconf dan automake.** Tool-tool standar yang diperlukan, masing-masing

berguna untuk mengkonfigurasi source code dan menggenerasi Makefile.in bergaya GNU.

- **Make. GNU Make.** Tool standar yang perlu Anda miliki untuk bekerja dengan Makefile.
- **Colorgcc.** Program ini berguna untuk mewarnai output dari gcc. Dengan demikian, output dari proses kompilasi akan lebih mudah terbaca. Error dan warning akan diwarnai dengan warna khusus.
- **lcmake.** Versi lain Make dengan gaya C.
- **distcc.** Proses kompilasi bisa berlangsung selama berjam-jam. Bahkan, bisa saja menjadi harian untuk proyek yang sangat besar. Sebuah komputer tentunya tidak cukup. Apabila terhubung ke jaringan dengan beberapa komputer, Anda dapat meminta komputer lain untuk membantu Anda. Tool distcc dapat Anda gunakan untuk itu. Dengan demikian, kompilasi tidak lagi dilakukan oleh komputer Anda sendiri, namun dibantu oleh beberapa komputer di jaringan.

Debugger

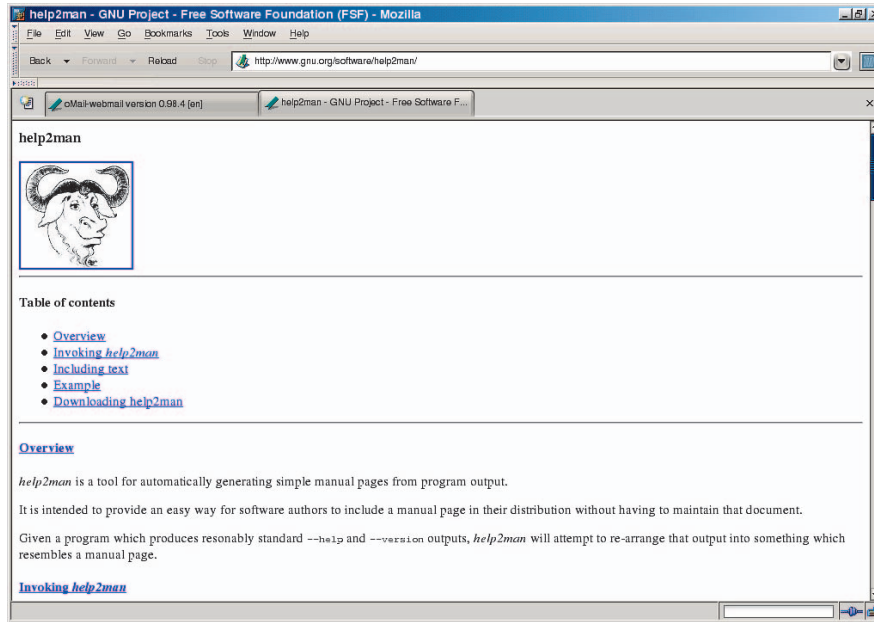
Proses debugging adalah proses yang

terkadang menyebalkan. Dengan berbagai tool berikut, proses debugging akan terasa lebih mudah dan seru.

- **ddd.** Data display debugger. GUI untuk gdb dan dbx. Struktur data bisa direpresentasikan dalam grafik dan ditampilkan secara interaktif. Program dapat didebug dalam C, C++, Pascal, Modula-2, Fortran, Ada dan assembly.
- **ElectricFence.** Malloc debugger untuk program C Anda.
- **Gdb.** Debugger GNU. Tool standar yang harus Anda miliki untuk debugging.
- **Insight.** GUI untuk gdb. Sangat matang dan dekat dengan gdb.
- **Kdbg.** GUI untuk gdb. Menyediakan user interface intuitif untuk pengaturan breakpoint, memeriksa variabel dan menjalankan bagian kode-kode Anda.
- **Memprof.** Memory profiler dengan interface GNOME. Dapat Anda gunakan untuk memeriksa memory leak.
- **Valgrind.** Memory management debugger. Tool pemeriksa memory yang telah sangat terkenal. Valgrind dapat memeriksa semua operasi memori. Sebagai contoh read, write, malloc, new, free, dan delete. Valgrind dapat menemukan penggunaan memori tak



▲ Anjuta.



▲ Help2man.

terpakai, akses ke memori yang telah dibebaskan, overflow, operasi stack ilegal, memory leak, dan berbagai new/malloc/free/delete ilegal. Datang juga dengan cachegrind, profiler yang dibangun di atas engine valgrind.

Document generator

Dokumentasi adalah komponen penting dalam pemaketan suatu program. Walaupun program yang ditulis tidak terlalu besar, dokumentasi tetaplah hal penting. Apalagi, jika aplikasi yang Anda tulis ternyata mengekspos API. Dokumentasi menjadi hal yang wajib di sini. Untuk membuat dokumentasi, berbagai tool bisa kita gunakan. Berikut ini adalah contoh-contohnya:

- **doxygen.** Doxygen adalah sistem dokumentasi untuk C, C++, Java dan IDL. Doxygen dapat menggenerasi class browser online (HTML) dan offline (LaTeX) dari source yang lengkap dengan dokumentasi. Dokumentasi akan diekstrak langsung dari source code.
- **Help2man.** Program ini dapat mengubah isi dari opsi `-help` menjadi manual.
- **Kdoc.** Program ini dapat menggenerasi dokumentasi online (HTML) dan offline (LaTeX) dari header C++.

IDE/Editor

Editor adalah tool yang selalu dibutuhkan

dalam penulisan program. Editor yang jelek akan mempersulit pembuatan program, sebaliknya, editor yang lengkap dan khusus digunakan sebagai alat bantu pemrograman akan membuat proses penulisan source code menjadi lebih menyenangkan. Selain editor, akan lebih baik apabila tersedia pula IDE. IDE, apalagi yang dibuat khusus untuk bahasa yang Anda gunakan benar-benar akan membuat proses penulisan source code menjadi lebih cepat. Berikut ini adalah beberapa contoh editor/IDE:

- **Anjuta.** IDE yang dimotori oleh programmer India ini memang luar biasa. Apalagi jika Anda menggunakan bahasa C, C++ atau GTK+/GNOME. Bahasa lain yang didukung juga cukup banyak. Beberapa fitur yang sangat berguna adalah code folding dan auto complete. Anjuta juga termasuk IDE yang sangat configurable. Hampir semua hal dapat diatur sesuai keinginan. Anjuta sangat memperhatikan preferensi programmer.
- **Eclipse.** Eclipse adalah IDE yang dapat dikembangkan dengan bantuan plugin. Programmer Java sangat cocok bekerja dengan IDE ini.
- **Eric.** Eric adalah IDE untuk bahasa Python, yang ditulis dengan bantuan pustaka PyQt. Sebagai IDE untuk Python, Eric sebenarnya tidak terlalu menyenangkan. Namun, kekhususannya untuk programmer Python menjadikan editor ini dapat mengerti kebutuhan dan gaya pemrograman python.
- **Vim.** Vim mungkin hanyalah editor. Namun, sebagai editor, fleksibilitas Vim tak tergantikan. Banyak programmer yang mengandalkan Vim untuk menulis source code mereka. Tak tanggung-tanggung, banyak diantara mereka yang mengerjakan aplikasi web kompleks dengan Vim. Bisa Anda bayangkan



▲ Valgrind.

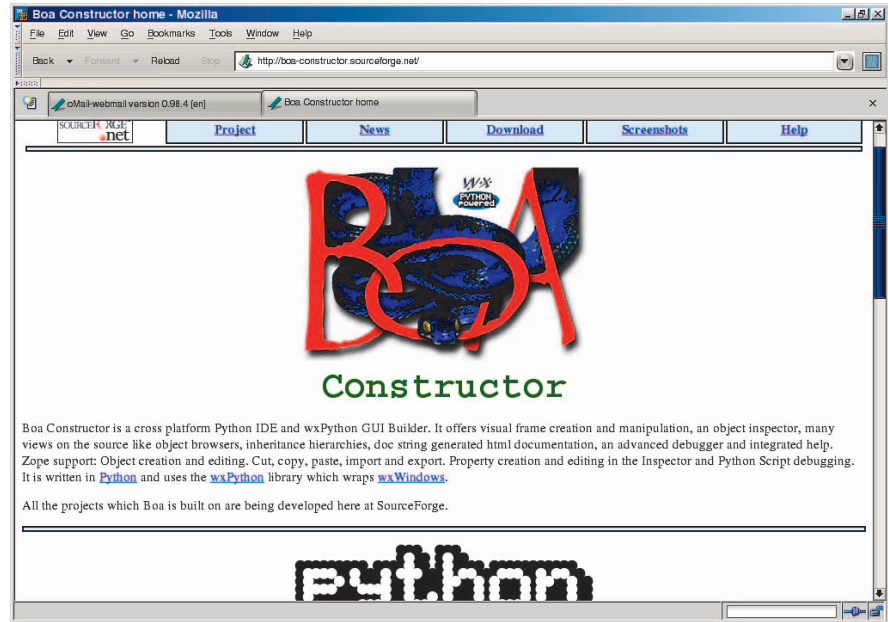
betapa banyaknya hal yang harus dilakukan.

- **Emacs.** Emacs ataupun Xemacs adalah editor komplik yang luar biasa. Mungkin user interfacenya konvensional. Namun, emacs sejak dahulu telah dikenal sebagai editor untuk developer dan administrator sistem.
- **Kdevelop.** Kdevelop adalah IDE untuk Anda yang bekerja dengan lingkungan pemrograman X11, QT dan KDE. Fasilitas yang datang bersamanya termasuk documentation browser, syntax highlighting dan lain-lain.

Diagram modeller

Seorang system analyst akan membutuhkan diagram modeller untuk mempermudah pemodelan sistem. Hasil dari analyst tersebut dapat digunakan sebagai panduan oleh programmer. Di era yang canggih ini, sudah bukan zamannya lagi untuk menggunakan kertas. Di Windows, berbagai diagram modeller canggih kita kenal. Di Linux, diagram modeller belum secanggih aplikasi Visio ataupun aplikasi-aplikasi dari Rational Software. Namun, kita memiliki beberapa diagram modeller yang cukup layak untuk diandalkan:

- **dia.** Dia sebenarnya dirancang untuk secanggih Visio. Saat ini dia dapat digunakan untuk menggambar berbagai



▲ Boa-constructor.

tipe diagram. Sebagai contoh, entity relationship, UML, SADT, flowchart, network diagram, dan sirkuit sederhana. Bentuk baru dapat dikembangkan dengan bantuan XML dan subset SVG untuk menggambar. Diagram dapat diekspor ke berbagai format grafik, dan dapat dicetak.

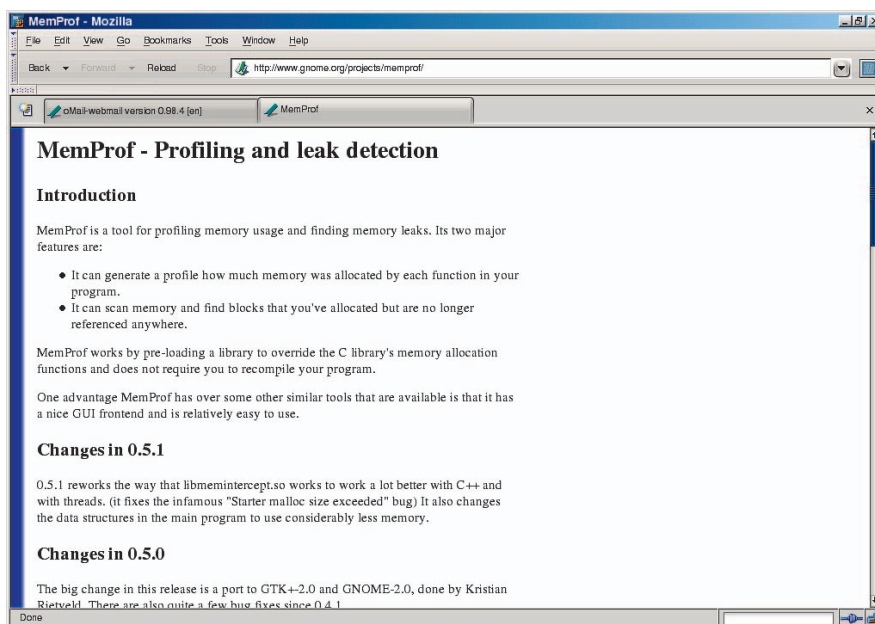
- **Umbrello.** Umbrello UML Modeller adalah alat bantu penggambaran UML untuk KDE. Saat ini, Umbrello termasuk salah satu aplikasi UML terbaik untuk

Linux. Bagi Anda yang sering bekerja dengan UML, maka program ini pantas Anda coba.

Version control

Seorang programmer yang bekerja sendirian mungkin tidak terlalu menganggap perlu version control. Semuanya dikerjakan oleh programmer tersebut. Mulai dari program utama, modul, dokumentasi sampai pernak-pernik lainnya. Setiap perubahan juga dikerjakan oleh sang programmer tersebut. Namun, tidak semua proyek dapat dikerjakan sendiri. Butuh kerja tim untuk mengerjakan proyek tersebut. Dan ketika bekerja dalam tim, perubahan sesuatu bukan menjadi masalah yang sederhana. Apa yang akan terjadi kalau file yang dibuat oleh A dapat dimodifikasi oleh B tanpa menyisakan aslinya? Untuk mencegah berbagai hal yang tak diinginkan dan menjadikan kerja tim lebih baik, maka revisi dan versioning adalah hal yang wajib. Terutama untuk menangani program, dokumentasi, grafik, dan paper. Berikut ini adalah contohnya:

- **CVS.** Yang satu ini merupakan tool standar dan termasuk di dalam distribusi-distribusi Linux.
- **Subversion.** Subversion melakukan apa yang dilakukan oleh CVS, hanya subversion datang dengan berbagai kelebihan.🐱



▲ Memprof.

Use the Battery!

Boleh dikatakan, Linux dan free software adalah surga bagi developer. Bagaimana tidak? Pustaka dan contoh tersebar di mana-mana. Tinggal comot, modifikasi kalau perlu, dan kembangkan aplikasi dengan bantuan pustaka tersebut. Beda jauh dengan di Windows. Developer di Windows umumnya perlu sedikit bekerja keras. Kembangkan sendiri, cari bajakannya, atau beli.

Istilah use the battery sangat umum ditemukan di dunia Python. Secara kasar dapat diartikan agar kita menggunakan apa yang telah tersedia daripada membuatnya sendiri. Tentunya, selama apa yang tersedia tersebut masih memenuhi kebutuhan kita.

Sebagai contoh. Anda ingin mengembangkan aplikasi MP3/OGG player. Apabila Anda ingin membuat semuanya sendiri dari nol, maka waktu yang diperlukan sangatlah banyak. Untuk mengerti format OGG yang terbuka saja butuh waktu yang sangat lama. Lebih bijak jika Anda mengembangkan player tersebut menggunakan pustaka yang telah disediakan. Dengan demikian, Anda dapat memfokuskan diri untuk menghasilkan program dengan tingkat usability tinggi.

Dalam konteks bahasa pemrograman, beberapa bahasa pemrograman bahkan datang dengan banyak pustaka atau modul. Berbagai fungsi telah disediakan. Kalau begitu, sangat disarankan agar Anda

menggunakan yang telah tersedia tersebut daripada membuatnya sendiri.

Menghemat cost!

Kenapa disarankan demikian? Jawabannya singkat. Menghemat cost. Waktu, uang, tenaga dan sebagainya. Membuat suatu pustaka atau fungsi bukanlah perkara mudah. Pustaka yang umum digunakan atau yang datang bersama bahasa pemrograman mungkin telah dikembangkan oleh berbagai pihak secara hati-hati. Mengulangi apa yang telah dikerjakan tersebut bukanlah hal yang bijak. Dan, belum tentu apa yang kita buat sendiri bisa lebih baik daripada yang telah tersedia.

Hal ini tentunya tidak berlaku umum untuk semua developer. Developer sistem yang bergerak di level bawah mungkin akan peduli untuk mengembangkan pustaka-pustaka baru. Atau, developer bahasa pemrograman tentu akan selalu memikirkan pustaka yang lebih baik lagi.

Baterai dan bahasa pemrograman

Python, Java, PHP dan Perl adalah contoh bahasa-bahasa pemrograman yang menawarkan berbagai pustaka siap pakai. Lihat saja modul-modul yang datang bersama Python. Berbagai pihak ketiga bahkan menawarkan tambahan modul untuk Python. Java datang dengan berbagai pustaka dan GUI toolkit sendiri.

Sementara, PHP berusaha untuk memenuhi kebutuhan penggunanya. Banyak sekali pustaka yang disertakan bersamanya. Developer Perl juga bertindak serupa. Perl datang berbagai pustaka siap pakai yang siap memanjakan penggunanya.

Hampir semua bahasa mencoba untuk datang selengkap mungkin. Namun, tidak ada satu pun bahasa yang datang dengan semua pustaka penting. Tidak perlu membuang waktu untuk mendapatkan bahasa terbaik dari sisi kelengkapan pustaka.

Berlaku juga untuk protokol

Ketika membuat aplikasi client server, kita bisa saja membuka socket dan melakukan semuanya sendiri. Sebagai contoh, apabila client mengirim 'xxx', maka kita membalasnya dengan mengirimkan 'yyy'. Dalam skala kecil, hal ini boleh-boleh saja. Himpunan perintah yang valid mungkin hanya 5 atau 10. Namun, sebenarnya, akan lebih bijak apabila kita mempergunakan protokol atau teknologi jaringan yang sudah matang daripada mengembangkan aturan sendiri.

Apabila Anda bekerja secara distributed, maka pertimbangkan untuk menggunakan Corba. XML-RPC juga sangat bagus untuk digunakan. Atau, Anda bahkan bisa menggunakan protokol HTTP atau FTP apabila kebutuhan Anda bisa dipenuhi dengan menggunakan protokol-protokol tersebut.

Pasar yang lebih besar

Siapa target pengguna aplikasi Anda? Apabila Anda membuat program, terutama utility atau tools, jangan batasi pengguna Anda. Perbanyak bahasa yang didukung, buatlah agar berjalan di platform lain, gunakan teknologi standar dan lengkapi dokumentasi!

Internationalization dan localization

Pembuatan program yang mendukung berbagai bahasa bukanlah pekerjaan yang harus dilakukan secara manual. Apabila aplikasi yang Anda buat hanya datang dengan dukungan bahasa Inggris, maka pasar Anda batasi untuk pengguna yang mengerti bahasa Inggris, atau pengguna yang mau berusaha untuk mengerti bahasa Inggris.

Dukungan i18n dan l10n adalah hal yang penting. Anda tidak perlu melakukannya sendiri. Sediakan saja frameworknya, dan mintalah bantuan komunitas untuk membantu. Percayalah, efeknya akan terasa sangat besar.

Multi platform

Linux adalah sistem operasi yang sudah tidak diragukan lagi kemampuannya. Tapi, apabila Anda adalah seorang software developer untuk aplikasi bisnis, maka membuat aplikasi yang hanya berjalan di Linux adalah tindakan konyol luar biasa. Memang makin banyak perusahaan yang menggunakan Linux. Namun, pengguna sistem operasi lain seperti Windows masih dominan.

Hal ini barangkali tidak dimengerti oleh beberapa programmer Visual Basic yang menganggap sistem operasi hanya ada satu di dunia ini. Mereka membuat aplikasi bisnis tanpa berpikir panjang. Perusahaan yang menggunakan software yang mereka buat harus menggunakan Windows. Perusahaan menjadi serba salah ketika ingin berpindah ke Linux. Ingin pindah, tapi ada aplikasi sejenis yang tidak berjalan di Linux. Porting jelas tidak mungkin. Emulasi juga bukan solusi.

Bagi developer, gunakanlah bahasa yang

mendukung multi platform. Bagi perusahaan atau konsumen, jangan pernah memiliki produk dari perusahaan yang hanya menawarkan dukungan untuk satu platform tertentu. Karena IT hanyalah sistem pendukung, jangan pernah memulai ketergantungan akan suatu teknologi.

Penggunaan teknologi standar

Sebagai contoh, Anda adalah perusahaan pembuat aplikasi office. Dalam membuat aplikasi office tersebut, Anda membuat pula format filenya sendiri, dan kemudian menutup format file tersebut. Yang mengerti hanyalah Anda, sang developer. Kian hari, client Anda kian bertambah.

Suatu hari, Anda merasa bahwa format file yang Anda gunakan rupanya memiliki kekurangan. Aplikasi office Anda pun diupgrade. Client-client Anda harus melakukan upgrade untuk dapat membaca format file baru tersebut.

Sebagai software developer, seberapa mampu Anda menjamin kelangsungan format file atau perusahaan Anda? Atau, bicara hal yang lebih ringan, seberapa yakin Anda dapat menjamin untuk tetap berjalan pada usaha yang sama?

Sebagai konsumen, seberapa yakin Anda akan perusahaan pembuat aplikasi office tersebut? Bagaimana kalau format filenya berubah terus (Anda harus membayar terus) atau perusahaannya bangkrut. Siapa yang akan menjamin data-data Anda?

Format file adalah salah satu contoh penggunaan teknologi standar. XML kini digunakan diantaranya sebagai standar format file. Berbagai aplikasi office populer seperti StarOffice, OpenOffice.org dan Koffice menggunakan XML untuk format filenya.

Bicara worst case. Andaikata Microsoft mengubah orientasi bisnisnya, atau suatu hari Microsoft Office sudah tidak dikembangkan dan didukung lagi. Hal ini memang sepertinya tidak masuk akal, tapi tetap saja mungkin. Sementara, sebagai pengguna, Anda telah menyimpan data

selama tahunan dengan salah satu versi Office-nya. Masalah seperti ini perlu senantiasa dicermati baik-baik.

Bagi software developer, gunakanlah teknologi standar. Bagi perusahaan atau konsumen, tanyakan seberapa standar teknologi yang digunakan oleh suatu produk sebelum membeli. Mintalah produk dengan teknologi standar agar ketika perusahaannya tidak lagi mendukung, Anda dengan mudah dapat meminta dukungan kepada pihak lain.

Terakhir, walaupun proyek yang dikembangkan tidaklah berskala besar, pertimbangkan selalu untuk datang dengan dokumentasi yang baik. Banyak konsumen menyukai dokumentasi yang lengkap. Pasar akan lebih besar.

Distribusi aplikasi

Distribusi adalah hal yang penting dan jangan dilupakan oleh software developer. Aplikasi skala besar tentu tidak hanya terdiri dari satu atau dua file. Dan terkadang, file pun tidak dapat dikopikan begitu saja.

Apabila target pengguna Anda adalah pengguna rumah atau pengguna mandiri, maka pastikan produk Anda dapat diinstall dengan mudah. Anda dapat memaketkannya dalam format paket-paket distribusi Linux seperti RPM, DEB dan lain sebagainya.

Apabila target pengguna adalah perusahaan dan harga produk telah termasuk support, maka kemudahan instalasi tidak terlalu penting. Namun, yang jelas, hal-hal yang bisa diotomatisasi jangan sampai dikerjakan secara manual.

Mengambil contoh StarOffice atau OpenOffice.org, Anda juga bisa mengembangkan sistem installer sendiri. Tentunya, waktu yang dibutuhkan untuk pengembangan akan lebih banyak.

Demikianlah pembahasan mengenai pengembangan aplikasi di Linux. Tentunya, tulisan ini tidak mungkin mencakup semua bahasa atau tool. Tidak ada maksud untuk mengunggulkan pihak tertentu atau menjatuhkan pihak lainnya. Selamat berkreasi dan sukses!