Mengelola Keuangan Pribadi dengan Linux

Untuk menjaga agar pengeluaran dalam sebulan dapat diketahui dengan jelas, kita dapat membuat laporan keuangan bulanan. Untuk memudahkan pembuatan laporan keuangan, sejumlah aplikasi keuangan yang terdapat di Linux, seperti GnuCash dan KMyMoney dapat digunakan untuk memenuhi kebutuhan ini.

Pendahuluan

Jika Anda ditanya, "Ada berapa uang di dompet Anda sekarang?", mungkin untuk menjawabnya mudah saja. Cukup buka dompet lalu hitung lembaran-lembaran uang yang ada di dalamnya.

Tapi kalau ditanya lebih lanjut pertanyaan-pertanyaan ini:

- Berapa pengeluaran Anda bulan terakhir? Atau rata-rata per bulan? Atau pengeluaran untuk makan, untuk anak (jika sudah memiliki anak), untuk pakaian, transpor, dan sebagainya?
- Berapa total kekayaan/aset yang Anda punya saat ini? Ini mencakup uang di dompet, tabungan, dan yang sedang dititipkan/dipinjamkan pada orang lain (piutang). Juga mencakup harta-harta yang tidak berupa uang, seperti kendaraan, tanah/rumah, dan sebagainya.
- Bagaimana dengan *net worth* Anda? Net worth adalah total aset dikurangi semua hutang Anda, jika ada.
- Bagaimana pertumbuhan net worth Anda dari bulan ke bulan, tahun ke tahun?
- Apakah Anda terlalu banyak menggunakan kartu kredit? Apakah pengeluaran Anda untuk kebutuhan yang tidak perlu (seperti liburan atau hobi) terlalu besar?

Mungkin menjawabnya tidak mudah atau bahkan tidak tahu. Padahal, buku-buku *self-help* atau praktisi penasihat keuangan mungkin akan mengatakan membiasakan mencatat semua kegiatan keuangan amat bermanfaat jika Anda ingin maju secara finansial. Bagaimana bisa mengontrol sesuatu (seperti pengeluaran) kalau tidak dicatat dan diketahui? Bagaimana bisa tahu apakah kita mundur atau maju secara finansial jika tidak ada angka patokan yang bisa dibandingkan dari waktu ke waktu?

Nah, untuk membantu menjawab pertanyaan-pertanyaan keuangan di atas dan mengelola keuangan pribadi Anda, di Linux tersedia program seperti GnuCash atau KMyMoney.

Kedua program yang disebutkan di atas termasuk dalam kategori pengatur keuangan pribadi atau program akunting pribadi. Kedua program ini fungsinya kurang lebih sama seperti Microsoft Money, Quicken, atau QuickBooks di Windows. Memang bisa saja kita menggunakan *spreadsheet* (seperti OpenOffice.org Calc, Gnumeric, Kspread, Excel) atau bahkan editor teks untuk melakukan pencatatan, tapi program keuangan pribadi memiliki fitur-fitur yang lebih spesifik dan mempermudah, seperti daftar akun yang sudah jadi, *interface* yang lebih mempercepat entri transaksi, laporan dalam bentuk teks dan grafik, bahkan berhubungan dengan fasilitas *Internet banking* untuk men-*download* transaksi (walau sayang, saat ini belum ada institusi finansial di Indonesia yang mendukung program-program keuangan pribadi seperti ini).

Di Linux ada berbagai program seputar keuangan/akunting, dari yang gratis/*open source* hingga berbayar, dari yang *desktop* hingga berbasis web, dari yang sederhana dan untuk pribadi sampai kompleks dan untuk perusahaan menengah. Sebutlah di antaranya Moneydance, Eqonomize!, MyPhpMoney, dan lain-lain.

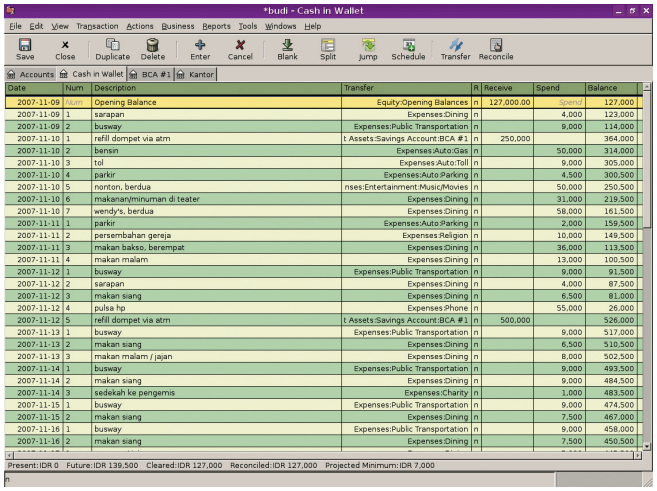
Namun, dalam artikel ini yang akan dibahas hanyalah dua yang telah disebutkan sebelumnya, GnuCash dan KMyMoney, karena keduanya sudah cukup lama ada dan paling populer. Artikel ini sendiri akan mengenalkan Anda bagaimana mulai menggunakan GnuCash dan KMyMoney untuk mengelola keuangan pribadi.

Langkah 1: Persiapan

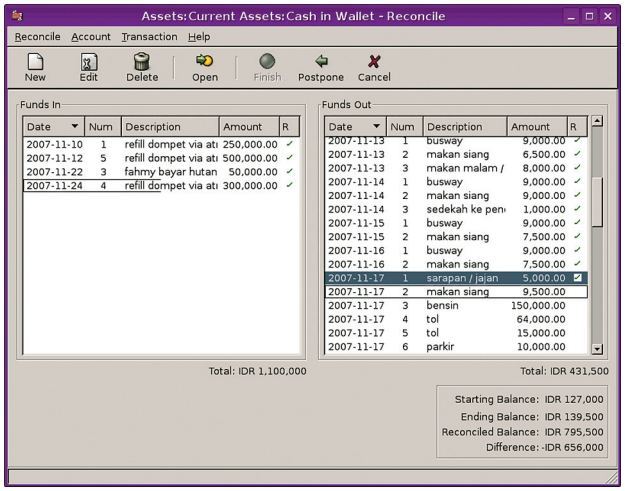
Langkah pertama dalam menggunakan program keuangan pribadi adalah mengumpulkan sumber data, yaitu transaksi sehari-hari. Pastikan semua transaksi yang Anda lakukan dicatat atau ada catatannya, agar nanti bisa dimasukkan ke program.

Setiap kali Anda belanja, simpanlah resinya. Untuk pengeluaran yang tidak ada resinya, seperti naik angkutan umum, parkir, atau memberi uang pada pengemis, catatlah di kertas/ponsel/PDA. Format catatan terserah, sebaiknya sederhana saja, mis: "3 jul: angkot 4000".

Jika Anda merasa repot mencatat pengeluaran yang banyak dan kecil-kecil seperti parkir, Anda bisa menyiapkan sejumlah uang di awal periode, mis: 40.000 di awal bulan. Di akhir periode, catatlah sisanya sehingga Anda mengetahui berapa pengeluaran total untuk periode tersebut.



Gambar 1. Register di GnuCash.



Gambar 2. Rekonsiliasi di GnuCash.

Setiap memperoleh penghasilan (gajian jika karyawan, menerima uang kiriman dari orang tua jika Anda mahasiswa, menerima hadiah/pemberian lain dalam bentuk uang), jangan lupa juga untuk dicatat.

Setiap menarik uang dari ATM, simpanlah *print-out* dari mesin ATM (atau catatlah jika tidak ada print-out).

Untuk lebih memberi gambaran, artikel ini akan menggunakan contoh seseorang fiktif yang sebutlah bernama Budi, pria, berusia 20-an, belum menikah, tinggal dan bekerja di Jakarta. Budi menyewa kamar kos, sehari-hari menggunakan transportasi umum ke kantor, walau kadang-kadang meminjam kendaraan milik kakaknya untuk bepergian. Ceritanya kita telah mengumpulkan catatan semua transaksi finansial Budi dari tanggal 9 Nov 2007 s.d. 30 Nov 2007 dan siap lanjut ke langkah berikutnya.

Sebagai catatan: file-file catatan keuangan Budi ini dapat di-*download di http://people.masterwebnet.com/steven/files/budi.zip*, atau dapat Anda temukan dalam bonus disc *InfoLINUX* edisi ini.

Langkah 2: Entri ke program GnuCash

Langkah kedua setelah data terkumpul dari "dokumen awal" (bon, resi, slip gaji) adalah memasukkannya ke program komputer. Memasukkan transaksi ke komputer dapat dilakukan sesering yang Anda suka, misalnya tiap hari, atau bisa juga seminggu atau sebulan sekali. Penulis sendiri, karena termasuk orang yang malas, biasanya melakukannya sekitar sebulan sekali, sembari melakukan rekonsiliasi (nanti di langkah 3). Perlu diakui, pekerjaan mengentri seperti ini bukanlah sesuatu yang amat mengasyikkan.

Bagian ini akan membahas program GnuCash dulu, KMyMoney akan dibahas berikutnya.

- **Setup awal**

  Jika GnuCash belum diinstal, installah dulu sesuai distro masing-masing (mis: dengan perintah 'apt-get install gnucash' jika di Debian). Lalu jalankan dengan mengklik entri menunya (biasanya di kategori Office) atau dengan mengetikkan 'gnucash' di shell.

  Pertama dijalankan, GnuCash lewat wizard-nya akan mengajak Anda membuat hirarki akun baru. Bagi yang buta akunting sama sekali, mungkin ada baiknya membaca-baca buku dasar akuntansi atau tutorial-tutorial yang ada di web, tapi cukuplah disebutkan di sini bahwa pada prinsipnya akuntasi berurusan dengan akun (rekening). Setiap kegiatan ekonomi dinyatakan dengan transaksi perpindahan uang dari satu akun ke akun-akun lainnya.

  Di wizard *New Account Hierarchy Setup*, pilihlah mata uang Rupiah (IDR), lalu klik *Forward*. GnuCash bisa membuatkan akun-akun spesifik tambahan misalnya fixed assets jika Anda memiliki kendaraan/rumah, akun pengeluaran istri/anak, dan lain-lain. Untuk sederhananya, cukup centang (✓) hanya *Common Accounts*. Klik *Forward*.

  Isilah saldo-saldo awal. Misalnya untuk dompet, ada di akun *Cash In Wallet* (di bawah Assets:Current Assets). Pada contoh kita, Budi pada awal pencatatan memiliki uang Rp127.000,- di dompet. Klik pada kolom *Opening Balances*, isikan 127000. Klik Forward, lalu Apply. GnuCash akan menampilkan tab Accounts berisi daftar akun yang kita punyai hasil setup tadi.

- **Membuat (sub)akun baru**

  Kita sebetulnya belum memasukkan semua aset dan akun yang kita punya. Jika Anda memiliki kendaraan atau rumah, ini juga sebetulnya perlu dimasukkan. Budi ceritanya memiliki juga satu akun tabungan BCA dan satu kartu kredit Citibank.

  Untuk membuat akun baru, bisa dengan klik kanan di bawah akun induk yang diinginkan. Contoh untuk tabungan BCA: klik kanan di bawah *Assets:Current Assets:Savings Account*, lalu pilih *New Account*. Ketikkan nama akun (bisa diganti-ganti nanti) mis: "BCA #1". Kita belum tahu saldo awalnya berapa, biarkan kosong dulu. Klik OK.

  Kartu kredit adalah rekening hutang (*liability*), jadi buatlah di bawah *Liabilities:Credit Card*. Saldonya juga ceritanya belum diketahui karena kita belum melihat tagihan bulanan, jadi kosongkan dulu.

  Membuat akun baru juga bisa sambil memasukkan transaksi, nanti akan diberikan contohnya.

  Selain membuat akun baru, untuk memudahkan bekerja dengan *locale Indonesia,* mari utak-atik sedikit menu *Edit > Preferences*. Pada tab Date/Time, Anda bisa memilih format Eropa dd.mm.yyyy atau sekalian ISO yyyy-mm-dd agar tidak bingung.

- **Memasukkan transaksi**

  Mari coba memasukkan transaksi, misalnya sarapan sebesar Rp4.000,- pada tanggal 9. Pertama-tama, di tab *Accounts*, dobel klik atau tekan *Enter* pada akun *Cash In Wallet*. GnuCash akan membuka tab baru yaitu register (jurnal) untuk akun ini. Di sana sudah ada 1 transaksi awal yaitu saldo awal (*Opening Balances*). Seperti telah dijelaskan, semua kegiatan ekonomi dinyatakan sebagai perpindahan uang dari akun ke akun. Saldo awal di dompet adalah modal yang kita berikan pada akun tersebut, karena itu dinyatakan sebagai perpindahan dari akun modal (*Equity:Opening Balances*) ke akun dompet. Akun dompet menerima (receive) sejumlah Rp127.000 dari akun modal.

  Untuk memasukkan transaksi sarapan kita, pertama ketikkan tanggal yaitu "09.09.2007", lalu tekan Tab. Ketikkan 1 sebagai nomor (ini adalah nomor urut transaksi pada hari tersebut, atau bisa juga nomor urut yang unik untuk semua hari, bergantung selera Anda), lalu tekan Tab. Ketikkan "sarapan" pada kolom *Description*, lalu tekan Tab. Pada kolom Transfer, kita memasukkan nama akun. Sarapan adalah pengeluaran, jadi pilihlah akun bernama "Expenses:Dining", lalu tekan Tab. Lewatlah kolom *Receive* dengan menekan Tab sekali lagi, karena dengan sarapan berarti kita mengeluarkan uang dari akun dompet ke akun pengeluaran makan. Pada kolom Spend, ketik 4000, lalu tekan Tab (atau Enter). Selesai. Anda telah memasukkan sebuah transaksi.

  Mari masukkan transaksi kedua, yaitu membayar ongkos *busway*. Karena tanggal masih sama, tekan Tab. Masukkan nomor urut 2 pada kolom Num. Masukkan "busway" pada kolom Deskripsi. Masukkan "Expenses:Public Transportation" pada kolom Transfer. Masukkan 9000 pada kolom Spend.

  Demikianlah cara memasukkan transaksi. Sederhana, bukan? Mari coba sekali lagi, yaitu transaksi mengambil uang dari akun tabungan via ATM pada tanggal 10. Transaksi ini adalah *Spend* bagi akun "Assets:Current Assets:Savings Account: BCA #1" dan Receive bagi akun dompet.

  Pertama tekan tombol "+". Ini akan shortcut untuk menambah 1 hari bagi nilai di kolom tanggal dari "09.09.2007" menjadi "10.09.2007". Untuk mengurangi 1 hari, tekan "-". Untuk menambah/mengurangi seminggu, tekan "+"/"-" sambil menekan Shift. Lalu tekan tombol Tab dan masukkan nomor urut 1. Ketik mis "ambil uang via ATM" atau "isi dompet dari ATM" di Deskripsi. Isikan nama akun BCA seperti disebutkan di atas dan isikan jumlah 250000 pada kolom Receive.

  Jika Anda berpindah ke tab Accounts GnuCash dan membuka akun BCA #1, maka di sana pun sudah akan ada catatan yang sama, yaitu Spend (bukan Receive) sebesar Rp250.000,- ke akun dompet. Inilah maksud dari pencatatan entri ganda (*double entry accounting*).

  Sebagai catatan: Anda juga sebetulnya bisa mengentri transaksi penarikan uang dari sisi register BCA #1.

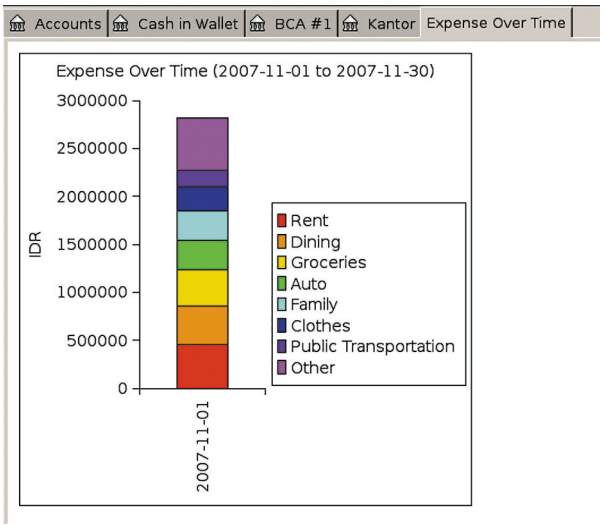
  Tip: GnuCash memiliki fasilitas *quickfill*. Jika Anda memasukkan transaksi yang deskripsinya sama, misalnya "busway" lagi pada tanggal 12 Nov, maka GnuCash akan mengisikan sisanya (akun, jumlah) untuk mempercepat entri. Anda tinggal mengubah jumlah uangnya, jika berbeda, misalnya. Lalu tekan Enter untuk memasukkan transaksi ini.

  Sampai di sini, sebaiknya Anda save file Anda. Pilih menu *File* > *Save* atau tekan Ctrl-S. Sering-seringlah men-save karena GnuCash tidak memiliki fasilitas autosave.

- **Membuat subakun baru sambil memasukkan transaksi**

  Memasukkan transaksi, seperti telah dijelaskan sebelumnya, amatlah mudah. Yang mudah-mudah susah adalah menentukan kategori atau akun. Kadang kategori tidak jelas dan bergantung pada masing-masing orang untuk menentukannya. Misalnya, jika Anda membeli beberapa pakaian untuk oleh-oleh sewaktu liburan, apakah termasuk dalam Expenses:Entertainment:Travel atau Expenses:Clothes? Silakan tentukan sendiri; yang penting adalah konsistensi.

  Jika Anda ingin mengawasi pengeluaran tertentu yang belum ada (sub)akunnya, buatlah subakun baru. Contoh, di GnuCash default-nya tidak ada akun khusus



Gambar 3. Laporan di GnuCash.

Gambar 4. Layar awal KMyMoney.

untuk mencatat pengeluaran jalan tol, yang ada hanyalah untuk parkir (Expenses:Auto:Parking), bensin (:Gas), perbaikan (:Repair and Maintenance), dan biaya lainnya (:Fees). Kalau Anda ingin bisa mentotal pengeluaran untuk tol saja, maka Anda bisa membuat subakun Expenses:Car:Toll misalnya. Atau bisa jadi Anda memiliki dua kendaraan dan ingin bisa melihat pengeluaran masing-masing. Anda bisa sedetil atau seumum mungkin dalam memilih akun/kategori, bergantung kepada tingkat obsesi Anda mencatat segala detail kehidupan pribadi.

Membuat subakun bisa dilakukan sambil memasukkan transaksi. Misalnya kita ingin memasukkan transaksi bayar tol tanggal 10 Nov. Di jurnal, masukkan tanggal, tekan Tab, ketik "tol", tekan Tab, lalu ketikkan "Expenses:Auto:Toll" di kolom Transfer. GnuCash akan memberi tahu bahwa akun ini belum ada dan menawarkan apakah ingin dibuat. Klik Yes.

- **Transaksi split**

  Kadang-kadang sebuah transaksi melibatkan lebih dari dua akun sekaligus. Berbelanja di supermarket adalah contoh transaksi "campur sari" seperti ini, karena biasanya Anda membeli bermacam-macam barang/kebutuhan. Pada contoh kita, Budi berbelanja di Carrefour pada tanggal 11 Nov untuk membeli makanan (beras, mi instan, sayur, buah) dan beberapa alat tulis (kertas, bolpen) menggunakan kartu kredit senilai Rp128.050,-. Untuk memasukkan transaksi split seperti ini, pertama bukalah akun kartu kredit Citibank dari tab *Accounts*. Kliklah tombol Split (atau pilih menu *Actions* > *Split Transaction*). Masukkan tanggal dan deskripsi. Di baris berikutnya, isikan charge 128050 untuk *Liabilities:Credit Card:Citibank* #1. Di baris berikutnya, isikan payment Rp 87.100 untuk akun *Expenses:Groceries*. Di baris berikutnya lagi, masukkan *payment* Rp39.950 untuk akun *Expenses:Supplies*. Baru tekan Enter.

  Untuk melihat detail split transaksi yang telah dimasukkan, pilih menu *View* > *Transaction Journal*. Untuk kembali ke tampilan semula, pilih menu *View* > *Basic Ledger*.

  Tip: jika Anda pemula, agar tidak bingung, untuk semua pembelanjaan di supermarket masukkan saja ke akun/kategori *Groceries*. Jika nanti Anda ingin melakukan *tracking* terhadap pengeluaran jenis barang tertentu, barulah buat sub-subakun di bawah Groceries misalnya. Katakanlah Anda ingin memilah-milah pengeluaran belanja ke dalam makanan/minuman, perlengkapan perawatan badan (sabun, sampo, pasta gigi, parfum), perlengkapan rumah tangga (deterjen, sabun cuci piring, pembersih lantai), dan lain-lain. Saya sendiri sempat membuat kategori pengeluaran cemilan (*snack*) karena ingin mengetahui apakah saya sudah terlalu banyak beli cemilan.

- **Piutang dan pengeluaran untuk pihak lain**

  Pada contoh kita, Budi berbelanja di Gramedia pada tanggal 21 November menggunakan kartu kredit sebesar Rp309.000,-. Sebagian buku yang dibeli adalah titipan dari teman, sebagian lagi untuk kantor. Untuk mencatat hal seperti ini, pertama masukkan transaksi pembeliannya seperti biasa (melibatkan akun-akun kartu kredit, *Expenses:Books*). Lalu buatlah akun baru *Assets:Current Assets:Receivables:Fahmy* untuk mencatat piutang terhadap teman yang bersangkutan, dan Assets:Current Assets:Receivables:Kantor untuk kantor. Lalu masukkan transaksi increase Rp50.000,- (pasangannya adalah Expenses:Books) untuk piutang Fahmy dan increase Rp60.000,- (pasangannya Expenses:Books) untuk piutang kantor. Efeknya adalah, hutang Anda bertambah Rp309.000,- karena berbelanja menggunakan kartu kredit. Tapi aset Anda juga bertambah Rp110.000,- karena jumlah uang ini akan bisa ditagih dari pihak yang berutang.

  Saat pihak yang berutang membayar, Anda tinggal buat transaksi kebalikan, yaitu decrease sehingga saldonya menjadi nol kembali. Contoh, Fahmy membayar hutangnya tanggal 22 Nov. Maka masukkan transaksi *decrease* Rp50.000,- pada akun piutang Fahmy (pasangannya adalah akun dompet). Efek akhir, saldo piutang Fahmy menjadi nol, dan uang di dompet bertambah Rp 50.000,-.

- **Pemasukan/Penghasilan**
  Untuk penghasilan, misalnya gajian yang didepositkan langsung ke akun tabungan, buatlah transaksi deposit ke tabungan dan pasangannya Income:Salary. Silakan lihat file GnuCash contoh yang disertakan di dalam file .zip yang saya sediakan.

Langkah 2: Entri Ke Program KMyMoney

Pada prinsipnya KMyMoney dan GnuCash memiliki fungsi yang kurang lebih sama, hanya saja KMyMoney lebih terinspirasi oleh program komersial Windows yaitu Microsoft Money. Selain konsep akun, KMyMoney juga memiliki konsep institusi keuangan, kategori pengeluaran, dan pihak terbayar (*payee*). Di GnuCash semua ini dinyatakan dalam (sub)akun.

Setup awal. Jika KMyMoney belum diinstal, installah dulu sesuai distro masing-masing (mis: dengan perintah 'apt-get install kmymoney2' jika di Debian). Lalu jalankan dengan mengklik entri menunya (biasanya di kategori Office) atau dengan mengetikkan 'kmymoney' di shell.

Buatlah file baru dengan menu File > New. Isikan informasi-informasi pribadi Anda seperti nama, alamat e-mail. Lalu pilihlah mata uang dasar Indonesian Rupiah. Lalu pilihlah template daftar akun en_US/, berhubung saat artikel ini ditulis belum ada yang membuat untuk Indonesia. Proses setup selesai.

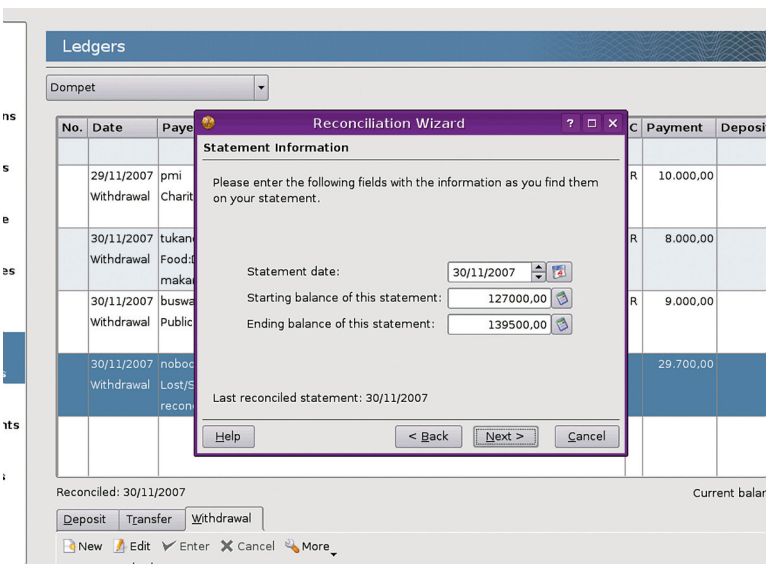
Dengan mengambil contoh yang sama (Budi), mari kita pertama-tama membuat akun tabungan dan kartu kredit. Di tab List akun, klik kanan pada Asset lalu pilih *New Account*. Lalu kliklah tombol *New Institution* dan isikan nama BCA dan tekan OK. Pilihlah institusi ini lalu klik *Next*. Pilihlah jenis akun *Savings* (tabungan), lalu Next. Isikan "BCA #1" sebagai nama akun, Next. Untuk nomor akun bisa diisikan bila Anda ingin, Next. Lalu isikan saldo awal lalu Next. Klik *Finish*. Untuk kartu kredit, prosesnya sama kecuali tipe akun adalah Credit Card, dan KMyMoney mewajibkan kita mengisi nama pihak terbayar dan jumlah kira-kira pembayaran kartu kredit untuk setiap bulannya. KMyMoney lalu otomatis akan menaruh akun ini di tempat yang sesuai, yaitu di bawah Liability.

Karena, tidak seperti di GnuCash, akun untuk dompet belum dibuatkan otomatis, maka kita buatlah juga akun baru bernama "Dompet" berjenis Cash, tanpa asosiasi institusi.

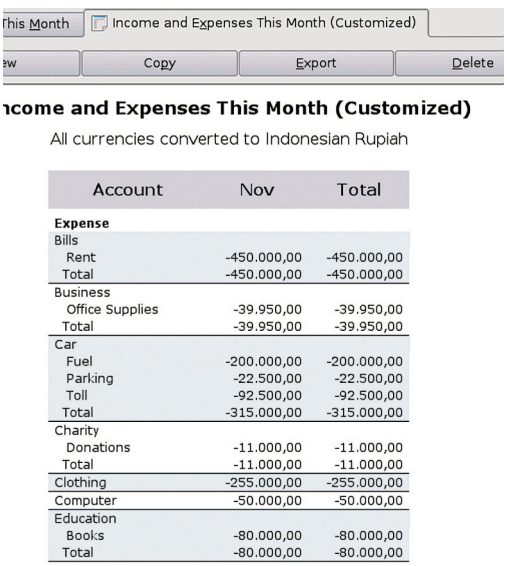
- **Memasukkan transaksi**
  Untuk memasukkan transaksi, misalnya sarapan tanggal 9 November, pertama dobel kliklah akun Dompet, maka Anda akan masuk ke *Ledger* (jurnal). Kliklah tab Withdrawal. Masukkan misalnya "tukang bubur" pada isian Pay To, dan pilih Food:Dining Out pada kategori. Set tanggal 9 Nov, masukkan 4000 pada jumlah, dan tekan Enter.
  Untuk transaksi penarikan uang via ATM dari akun tabungan, pertama masuklah ke ledger untuk akun tabungan BCA, lalu pilihlah tab Transfer. Isikan Pay To misalnya nama diri sendiri, "budi", lalu pilih akun Dompet untuk isian Transfer To.
  Tip: di KMyMoney pun Anda dapat menekan tombol "+" atau "-" untuk menambah-kurangi tanggal dengan kelipatan 1 hari.
  Jangan lupa untuk sering-sering men-save pula dengan Ctrl-S atau klik tombol Save. Tidak ada autosave di KMyMoney.

- **Membuat kategori baru**
  Untuk membuat kategori baru, bisa sambil memasukkan entri transaksi, atau di layar Categories. Misalnya untuk transaksi ongkos busway, saat mengentri withdrawal dari Dompet, isikan Public Transportation di kategori dan KMyMoney akan mengonfirmasi pembuatan kategori baru.

- **Transaksi split**
  Transaksi split adalah transaksi dengan lebih dari 1 kategori sekaligus. Contohnya adalah berbelanja untuk makanan dan alat tulis di supermarket. Untuk memasukkannya, pertama isikan Pay To, mis: "carrefour", lalu jumlah total mis: 128050. Baru klik tombol Split. Masukkan setiap kategori yang ada, mis: Food: Groceries sebesar 88.100 (lalu tekan enter atau klik tombol tanda centang (✓) berwarna hijau). Demikian pula untuk



Gambar 5. Rekonsiliasi di KMyMoney.



Gambar 6. Laporan di KMyMoney.

alat tulis, mis: di kategori *Business:Office Supplies* sebesar 39.950, dan tekan Enter. Baru klik Finish. Setelah itu, Anda perlu mengklik lagi tombol Enter, jika tidak maka daftar kategori yang tadi Anda masukkan akan hilang!

- **Piutang**

  Untuk mencatat pembelanjaan yang merupakan hutang orang lain, caranya mirip seperti di GnuCash. Pertama kita buat dulu akun piutang, mis: Piutang Fahmy dan Piutang Kantor. Agar mudahnya, buat tipenya *Cash* saja dan bukan *Loan*, karena untuk Loan ada perhitungan bunga periodik. Setelah itu masukkan transaksi pembelanjaannya. Lalu masukkan pula Deposit untuk akun-akun piutang.

  Saat pembayaran hutang, lakukan perpindahan dana (Transfer) dari akun piutang ke akun Dompet, sehingga piutang menjadi nol dan uang Anda bertambah. Lihat file KMyMoney yang disertakan dalam file .zip untuk detailnya.

Langkah 3: Rekonsiliasi di GnuCash

Setelah memasukkan semua transaksi keuangan, langkah berikutnya adalah rekonsiliasi. Rekonsiliasi adalah proses pencocokan catatan yang kita buat di program dengan "kenyataan yang sebenarnya", dengan kata lain, proses koreksi. Karena mungkin saja kita kelupaan memasukkan beberapa transaksi, melakukan salah ketik, dsb. Untuk akun-akun yang dimaintain oleh institusi keuangan, seperti tabungan bank atau kartu kredit, kita mencocokkan catatan kita dengan catatan dari bank. Untuk akun yang tidak ada institusi keuangannya, seperti uang di dompet, kita mencocokkan saldo terakhir catatan kita dengan uang yang ada di dompet. Rekonsiliasi biasanya dilakukan akhir bulan, atau saat kita menerima laporan/statement kartu kredit, tabungan, dan-lain-lain.

Untuk melakukan rekonsiliasi akun dompet di GnuCash pertama buka register dompet. Lalu pilih menu *Actions* > *Reconcile*. Misalnya, pada contoh kita, Budi melakukan rekonsiliasi akun dompet pada tanggal 30 November. Masukkan tanggal 30 November. Lalu hitunglah uang di dompet saat ini ada berapa, ternyata Rp147.000,- (padahal menurut catatan, Rp176.700,-). GnuCash lalu akan menampilkan daftar transaksi yang belum direkonsiliasi, untuk diperiksa ulang. Centanglah (✓) semua transaksi yang dirasakan benar. Anda bisa menghapus, mengedit, atau menambah transaksi. Pada contoh, ternyata memang tidak diketahui selisih yang ada sebesar Rp29.700,- itu ke mana, mungkin uangnya hilang, tercecer, atau dicuri, atau dibelikan sesuatu yang sudah dicoba diingat-ingat atau dicari bonnya tapi tidak ada. Karena itu kita mengakuinya saja saja sebagai uang yang hilang, di akun *Expenses:Adjustment* atau *Expenses:Lost/Stolen*. Buat transaksi tersebut, maka kini sudah tidak ada selisih lagi. klik tombol *Finish*. Kini Anda akan melihat bahwa di kolom bertitel "R" di tiap entri transaksi Anda akan melihat simbol "y" dan bukan "n" lagi, yang berarti bahwa transaksi tersebut statusnya telah direkonsiliasi.

Demikian juga untuk akun lain seperti tabungan BCA dan Citibank. Saat menerima tagihan kartu kredit atau mencetak mutasi di buku tabungan, lakukanlah rekonsiliasi. Untuk tabungan BCA pada contoh kita, kita belum memasukkan saldo awal. Masukkan dulu saldo awal untuk tanggal 1 Nov.

Setelah rekonsiliasi, kini kita siap membuat laporan (Langkah 4).

Langkah 3: Rekonsiliasi di KMyMoney

Untuk melakukan rekonsiliasi di KMyMoney, pilih menu *Account* > *Reconcile*. Klik Next. Masukkan saldo awal, saldo akhir, dan tanggal catatan sebenarnya (tanggal dari tagihan kartu kredit, cetak buku tabungan, atau pengecekan saldo dompet). KMyMoney juga mengizinkan Anda memasukkan bunga dan pajak bunga/biaya administrasi (yang umumnya selalu ada setiap bulan pada akun tabungan) agar bisa dimasukkan otomatis.

Setelah itu Anda melakukan pencentangan (✓) pada masing-masing transaksi yang sudah dicek langsung pada register akun ybs (berbeda dengan GnuCash yang menggunakan *window* terpisah). Jika masih ada selisih, Anda bisa mengedit atau menambahkan transaksi. Setelah proses selesai, entri yang telah direkonsiliasi akan dilabeli "R" pada kolom "C", dan KMyMoney akan memberi peringatan agar tidak mengedit kembali transaksi yang telah direkonsiliasi.

Langkah 4: Laporan

Sekarang setelah mencatat semua data, kita dapat melihat statistik dan laporannya.

- **Total pengeluaran**

  Untuk sekadar melihat total saldo tiap akun, Anda dapat melihat di tab Accounts. Sayangnya, di KMyMoney defaultnya saldo sebuah kategori pengeluaran tidak mencakup saldo dari subkategorinya. Misalnya untuk Car, di mana terdapat subkategori Fuel, Toll, Parking. Total pengeluaran untuk Car tidak mencakup sub-subkategorinya. Saya belum tahu apakah ada setting di KMyMoney untuk mengubah hal ini.

  Misalnya, pada contoh kita, ternyata Budi mengalami "besar pasak daripada tiang", alias total pengeluaran sebesar Rp2.818.265,- lebih besar dari penghasilan Rp2.500.000,-. Saldo tabungan memang bertambah di akhir bulan dibandingkan awal bulan, tapi masih ada tagihan kartu kredit yang harus dibayar. Dari sini dapat dianalisis lebih lanjut di mana pengeluaran terlalu banyak dan apa yang bisa dikurangi.

- **Laporan**

  Kedua program menyediakan berbagai laporan, walaupun KMyMoney belum menyertakan grafik. Contoh laporan: pengeluaran dan pemasukan per bulan dan per tahun, perkembangan saldo tabungan, total aset, net worth per bulan dan per tahun, aliran kas, dan lain-lain.

  Berbekal laporan ini dan patokan-patokan budget (mis: untuk kebutuhan sehari-hari jangan lebih dari 30% dari penghasilan, perlu ada menabung sekitar 10-20% dari penghasilan, untuk berlibur dalam setahun jangan lebih dari gaji sebulan, dan lain-lain; lebih banyak lagi mengenai ini bisa Anda dapatkan dari buku-buku atau konsultasi dengan penasihat keuangan), Anda bisa menganalisis masalah dan lebih mengendalikan keuangan pribadi dan keluarga. Selamat mencoba!■

Steven Haryanto [steven@masterwebnet.com]

Membangun Aplikasi yang Mendukung Plug-in

Program-program komputer dewasa ini umumnya datang dengan fasilitas *plug-in*—apapun nama yang digunakan—yang dapat digunakan untuk menambahkan fungsi tertentu pada aplikasi. Tak jarang, plug-in-plug-in tersebut bisa dikembangkan pula oleh pihak lain. Seru, bukan?

Sebutlah berbagai program berikut: GIMP, Firefox, XMMS, dan OpenOffice.org. Mereka merupakan aplikasi besar yang dapat dikembangkan lebih lanjut dengan mekanisme plug-in atau extension, atau macro, dan lainnya. Apapun namanya, bagaimanapun bentuknya, fungsi dari plug-in umumnya adalah mengembangkan fungsi utama aplikasi. Sebagai contoh, plug-in pada GIMP dapat digunakan untuk memproses gambar dengan algoritma tambahan, yang belum dimiliki oleh GIMP. Atau, plug-in pada XMMS dapat digunakan untuk membuka format file yang belum dikenal oleh XMMS. Atau, extension pada Firefox bisa digunakan untuk mengatur proses browsing, dan lainnya. Bahkan, plug-in pada game dapat pula digunakan untuk mengatur artificial intelligence game tersebut.

Ketika kita membangun program skala besar, di mana user bisa menyesuaikan solusi yang kita berikan, akan lebih mudah apabila kita memanfaatkan plug-in.

Sebagai developer, kitalah yang memutuskan bagaimana mekanisme plug-in diterapkan. Beberapa pertimbangan:

- Bagaimana scope plug-in yang akan kita bangun. Apakah plugin dapat mengakses core program?
- Fungsi-fungsi apa saja yang bisa dilempar ke plug-in?
- Bagaimana metode I/O yang harus dilakukan oleh plug-in?
- Format plug-in seperti apa yang bisa kita gunakan?
- Apakah plug-in bisa dikembangkan dengan mudah oleh developer yang tidak berhubungan dekat dengan developer program utama?

Di tulisan ini, kita akan membahas berbagai contoh program yang memanfaatkan plug-in untuk mengembangkan fungsionalitas program. Program utama, atau bisa kita sebut sebagai host, dibangun dengan bahasa C, dan semua plugin yang ada, dibangun dengan bahasa lua.

Untuk informasi tentang lua, kunjungilah website www.lua.org. Lua, dalam hal ini, dipilih karena berukuran cukup kecil, cukup fleksibel untuk digunakan dari C dan cukup kaya fitur. Untuk bertukar data, stack digunakan. Untuk mendapatkan nilai dari lua: kita menjalankan lua, yang akan mempush data ke stack. Untuk meng-assign nilai tertentu untuk variable lua: kita push nilai ke stack dan menjalankan lua (yang akan pop nilai tersebut).

Semua contoh di tulisan ini dibangun di atas sistem Slackware 11, dengan GCC versi 3.4.6 dan lua versi 5.1.2. Sebagai catatan, untuk mencoba, pastikan lua yang terinstal adalah lua 5.1.x. Semua contoh dilisensikan GPL.

Kompilasi bisa dilakukan dengan pola perintah:

```
$ gcc -o <out> <src>.c -I/usr/include/lua/5.1 -L/usr/lib/lua/5.1 -llua
```

Fungsionalitas yang disediakan

Sebagai developer, pastikan kita mengetahui benar fungsionalitas apa yang akan disediakan oleh sebuah plug-in. Tentunya, hal ini harus disesuaikan dengan program yang akan kita bangun. Mari berandai-andai. Katakanlah kita ingin membangun program A, sebuah program gambar sederhana. Sejatinya, program utama kita tahu persis bagaimana menggambar di atas kanvas dengan segala pernak-perniknya. Yang program kita tidak tahu adalah, bentuk apa yang harus digambar.

Sebagai sebuah program penggambar, bentuk tentu diserahkan kepada *user*. Userlah yang menggerakkan pensil dan membuat bentuk yang diinginkan. *Nah*, berhubung tidak setiap user bisa menggambar dengan baik dan cepat, alangkah baiknya kalau kita sediakan fasilitas semacam *predefined-shape,* seperti menggambar kotak, lingkaran, sampai bentuk-bentuk lainnya.

Dalam hal ini, adalah sangat mungkin untuk memanfaatkan plug-in, di mana plug-in kita menyediakan algoritma penggambaran. Sebagai developer program utama, kita tinggal melempar parameter apa yang dibutuhkan oleh plugin dan menerima algoritma penggambaran.

Tidak berhenti sampai di sana, tentunya, plugin bisa pula dikembangkan

IKLAN

sebagai penyedia format file (penyimpanan dan pembukaan) ataupun sebagai *tool* tambahan yang bisa memperkaya program.

I/O dan user interface

Umumnya, plug-in akan membutuhkan parameter tertentu dan dapat menghasilkan output tertentu (I/O).

Untuk itu, setidaknya kita bisa membaginya menjadi dua cara:

1. I/O semua dilakukan oleh program utama. Program utama kemudian melemparnya ke plug-in. *Output* dari plug-in juga diterima oleh program utama, yang kemudian menyajikannya ke user. Plug-in, dalam hal ini, tidak berhubungan langsung dari sisi I/O dengan user.
2. I/O bisa pula dikerjakan oleh plugin. Baik untuk program *text-based* ataupun GUI. Hanya, perhatikan konsistensi UI. Agar plug-in mudah dikembangkan, kita harus menyiapkan struktur data yang baik untuk I/O, yang bisa dibaca dan dikerjakan oleh plug-in.

Program utama juga harus memikirkan bagaimana caranya agar registrasi komponen UI bisa dilakukan dengan mudah dan fleksibel.

Format plug-in

Di tulisan ini, kita memilih untuk menggunakan lua. Dengan demikian, untuk membuat plugin, seorang developer harus mengerti aturan plugin kita dan bahasa lua.

Lua adalah satu pilihan. Tapi, Anda bisa memilih banyak cara. Di antaranya:

- Menggunakan *shared object* atau *platform native*. Di Linux, ini berupa file .so. Tidaklah mudah untuk membangun host dan plug-in dengan C. Cara ini sekilas pernah kita bahas pada pembahasan pembuatan *shared library*.
- Menggunakan *script*, seperti lua, python, dan lainnya.
- Menggunakan format yang kita mengerti sendiri. Lantas, kita buat juga program semacam 'plug-in development kit' yang akan digunakan oleh pembuat plug-in. Cara yang ketiga ini sangat rumit apabila Anda ingin membangun format yang aman, kompatibel antarversi, plus dilengkapi dengan fasilitas script.

Perhatikan juga developer plug-in

Siapa saja yang boleh membangun plug-in untuk program Anda? Perusahaan Anda sendiri, *partner* yang menandatangani non-*disclosure-agreement*, atau siapapun?

Kalau hanya Anda atau tim *development*, ini mungkin relatif mudah, dalam arti bagaimana program utama dan plug-in berkomunikasi.

Tapi, kalau siapa saja bisa membangun plug-in untuk program Anda, maka cara berkomunikasi harus sangat fleksibel. Idealnya, program tinggal membaca plug-in Anda dan mengerti bagaimana harus bertindak. Semua metadata tersedia di plug-in dan program utama tahu bagaimana mengartikannya. I/O juga sudah diatur. Akses ke UI juga sudah beres. Kalau Anda menggunakan format sendiri, pastikan tersedia development kit yang memadai.

Program 1: plug-in supersederhana

Di contoh pertama ini, kita akan membangun program 1, dengan source code 1.c, yang akan memanggil plugin 1.lua. Plugin 1.lua ini akan menampilkan tulisan ke stdout. Kesemua file tersebut disimpan dalam direktori yang sama.

Berikut ini adalah *source code* 1.c:

```
#include <stdio.h>
#include <lua.h>
#include <lauxlib.h>

int main(void)
{

  lua_State *L;
  L = lua_open();
  luaL_openlibs(L);

  fprintf (stdout, "[host] Selamat
datang di C\n");

  luaL_loadfile(L,"1.lua");
  lua_call (L, 0, 0);

  lua_close (L);

  fprintf (stdout, "[host] Kembali
ke C\n");

  return 0;
};
```

Berikut ini adalah isi script 1.lua:

```
print ("[plugin] Hello World")
```

Penjelasan C:

- Untuk menggunakan lua, kita menggunakan header lua.h dan lauxlib.h (perhatikan penulisannya, jangan sampai salah dituliskan menjadi luaxlib.h)
- Pertama-tama, kita mendefinisikan lua state, membuka lua state dan membuka semua pustaka yang dibutuhkan:

```
lua_State *L;
L = lua_open();
luaL_openlibs(L);
```

- Pencetakan kita lakukan seperti biasa:

```
fprintf (stdout, "[host] Selamat
datang di C\n");
```

- Selanjutnya, kita membuka script 1.lua dan menjalankan script tersebut:

```
luaL_loadfile(L,"1.lua");
lua_call (L, 0, 0);
```

- Setelah itu, kita tutup lua state:

```
lua_close (L);
```

- Dan, melakukannya pencetakan seperti biasa:

```
fprintf (stdout, "[host] Kembali
ke C\n");
```

Penjelasan lua:

- fungsi print() dapat digunakan untuk menampilkan string ke stdout.

Sekarang, user bisa memodifikasi file 1.lua dan melakukan apapun, dan ketika 1 dijalankan, user akan mendapatkan apa yang telah diubah di 1.lua. Source code 1.c tidak perlu dikompilasi ulang.

Dengan menggunakan lua, *executable* kita membutuhkan pustaka tambahan, yaitu liblua.so.

Program 2: Mengirim data ke plug-in

Di contoh kedua ini, kita akan membangun program 2, dengan source code 2.c dan plugin 2.lua. Pastikan semua file berada di dalam direktori yang sama. Berbeda dengan contoh pertama, ketika dijalankan, kita akan mengirim data ke plugin (2.lua), sehingga plug-in menampilkan string sesuai dengan apa yang kita kirimkan.

Berikut ini adalah source code 2.c:

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

int main(int argc, char * argv[])
{
```

```
  char  nama[255];
  lua_State *L;

  if (argc != 2)
  {
   fprintf (stderr, usage: %s
<name>\n, argv[0]);
   return 1;
  }

  L = lua_open();
  luaL_openlibs(L);

  fprintf (stdout, [host] Selamat
datang di C\n);

  luaL_loadfile(L,2.lua);
  strcpy (nama, argv[1]);
  lua_pushstring (L, nama);
  lua_setfield (L, LUA_GLOBALSINDEX,
myname);
  lua_call (L, 0, 0);
  lua_close (L);;
  fprintf (stdout, [host] Kembali
ke C\n);
  return 0;
};
```

Berikut ini adalah isi script 2.lua:

```
print ([plugin] Hello  .. myname
..  apa kabar?)
```

Contoh output:

```
$ ./2 NOP
[host] Selamat datang di C
[plugin] Hello NOP apa kabar?
[host] Kembali ke C
```

Penjelasan C:

- Lihatlah juga pembahasan 1.c
- Di contoh ini, kita mendapatkan nama dari argv[1], push string tersebut ke stack lua dan mengatur nilai tersebut sebagai isi variable global myname, dan kemudian menjalankan script.

```
strcpy (nama, argv[1]);
lua_pushstring (L, nama);
lua_setfield (L, LUA_GLOBALSINDEX,
myname);
lua_call (L, 0, 0);
```

Penjelasan Lua:

- operator .. digunakan untuk menggabungkan string
- script akan mencetak sebaris string sesuai dengan isi variable myname, yang kita kirimkan sebelumnya

Program 3: Menggunakan pustaka standar plug-in

Kita akan membangun program 3, dengan source 3.c dan plugin 3.lua. Contoh ini mirip dengan contoh 1, namun mendemonstrasikan bagaimana kita memanfaatkan pustaka yang dimiliki oleh Lua, dalam hal ini mendapatkan informasi waktu.

Berikut ini adalah source code 3.c:

```
#include <stdio.h>
#include <lua.h>
#include <lauxlib.h>

int main(int argc, char * argv[])
{
  lua_State *L;
  L = lua_open();
  luaL_openlibs(L);

  fprintf (stdout, [host] Selamat
datang di C\n);
  fprintf (stdout, [host] Tanggal
dan jam akan didapatkan dari plugin\
n);

  luaL_loadfile(L,3.lua);
  lua_call (L, 0, 0);
  lua_close (L);;
  fprintf (stdout, [host] Kembali
ke C\n);
  return 0;
};
```

Berikut ini adalah isi script 3.lua:

```
print ('[plugin]')
d=os.date('*t')
for k,v in pairs(d)
do
  print (k,v)
end
```

Penjelasan Lua:

- Kita dapatkan informasi waktu dengan pustaka os, fungsi date().
- Selanjutnya, kita melakukan iterasi table dengan fungsi pairs().

Program 4: Contoh sederhana metadata plugin

Kita akan mulai menyusun format plug-in kita sendiri, yang kita mulai dengan informasi sederhana tentang plug-in, seperti nama, pembuat, *copyright*, dan lisensi. Di program 4 ini, kita akan membaca informasi tersebut dan menampilkannya di stdout. File plugin disimpan di 4.lua.

Berikut ini adalah source code 4.c:

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

int main(int argc, char * argv[])
{
  char plugin_name[255];
  char plugin_author[255];
  char plugin_copyright_string[255];
  char plugin_license[16];

  lua_State *L;
  L = lua_open();
  luaL_openlibs(L);

  fprintf (stdout, [host] Selamat
datang di C\n);
  fprintf (stdout, [host]
Mendapatkan properti plugin\n);

  luaL_loadfile(L,4.lua);
  lua_call (L,0,0);

  lua_getfield (L,LUA_GLOBALSINDEX,
plugin_name);
  lua_getfield (L,LUA_GLOBALSINDEX,
plugin_author);
  lua_getfield (L,LUA_GLOBALSINDEX,
plugin_copyright_string);
  lua_getfield (L,LUA_GLOBALSINDEX,
plugin_license);

  strcpy (plugin_name, lua_
tostring(L,-4));
  strcpy (plugin_author, lua_
tostring(L,-3));
  strcpy (plugin_copyright_string,
lua_tostring(L,-2));
  strcpy (plugin_license, lua_
tostring(L,-1));

  fprintf (stdout, \t[host] Nama
plugin: %s\n, plugin_name);
  fprintf (stdout, \t[host] Pembuat
plugin: %s\n, plugin_author);
  fprintf (stdout, \t[host]
Copyright plugin: %s\n, plugin_
copyright_string);
  fprintf (stdout, \t[host] Lisensi
plugin: %s\n, plugin_license);

  lua_close (L);;
  fprintf (stdout, [host] Kembali
```

```
ke C\n);
  return 0;
};
```

Berikut ini adalah isi script 4.lua:

```
--plugin untuk aplikasi 4
plugin_name = /etc/passwd parser
plugin_author = Noprianto
plugin_copyright_string = (c)
Noprianto, 2007
plugin_license = GPL
```

Penjelasan C:

- Kita membuka file 4.lua, menjalankan dan mendapatkan 4 informasi dari plugin (lua_getfield), dan meng-assign nilainya ke 4 variable di C (strcpy).

Program 5: Menjalankan fungsi di plug-in

Kita melangkah maju. Di program 5 ini, selain mendapatkan informasi dari file plug-in (5-1.lua, 5-2.lua dan 5-3.lua), kita juga menjalankan fungsi plugin_main di setiap file plugin tersebut. Saat ini, isi plugin_main dari file-file plugin tersebut hanyalah mencetak ke stdout.

Berikut ini adalah source code 5.c:

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

int main(int argc, char * argv[])
{
  char temp[255];
  char plugin_name_temp[255];
  int i;
  fprintf (stdout, "[host] Selamat
datang di C\n");
  fprintf (stdout, "[host] Simulasi
mendaftarkan menu dan menjalankan
fungsi utama plugin\n");

  for (i=1; i<=3; i++)
  {
   sprintf (temp, "5-%d.lua", i);

   lua_State *L;

   L = lua_open();
   luaL_openlibs(L);

   luaL_loadfile(L,temp);

   lua_call (L,0,0);

   lua_getfield (L,LUA_GLOBALSINDEX,
"plugin_name");
   strcpy (plugin_name_temp, lua_
tostring(L,-1));
   fprintf (stdout, \n\n[host]
Mendaftarkan menu plugin baru: %s\
n, plugin_name_temp);
   fprintf (stdout, Menjalankan
plugin %s\n, plugin_name_temp);
   lua_getfield (L, LUA_GLOBALSINDEX,
plugin_main);
   lua_call (L,0,0);
   fprintf (stdout, Selesai
Menjalankan plugin %s\n, plugin_
name_temp);
   lua_close (L);;
  }
  fprintf (stdout, [host] Kembali
ke C\n");
  return 0;
};
```

Berikut ini adalah isi script 5-1.lua:

```
--plugin untuk aplikasi 5
plugin_name = /etc/passwd parser
plugin_author = Noprianto
plugin_copyright_string = (c)
Noprianto, 2007
plugin_license = GPL

function plugin_main()
  print (Selamat datang di  ..
plugin_name)
end
```

Berikut ini adalah isi script 5-2.lua:

```
--plugin untuk aplikasi 5
plugin_name = /etc/group parser
plugin_author = Noprianto
plugin_copyright_string = (c)
Noprianto, 2007
plugin_license = GPL

function plugin_main()
  print (Selamat datang di  ..
plugin_name)
end
```

Berikut ini adalah isi script 5-3.lua:

```
--plugin untuk aplikasi 5
plugin_name = /etc/fstab parser
plugin_author = Noprianto
plugin_copyright_string = (c)
Noprianto, 2007
plugin_license = GPL

function plugin_main()
  print (Selamat datang di  ..
plugin_name)
end
```

Penjelasan C:

Untuk menjalankan fungsi di Lua dari C, secara umum kita melakukan serangkaian tugas berikut: push nama fungsi, push argumen fungsi (dalam hal ini belum ada), jalankan lua_call atau lua_pcall, pop hasil dari stack.

Program 6: I/O oleh plugin

Contoh Program 6 membahas tentang I/O, namun dilakukan dari plug-in (6.lua). Dengan demikian, program utama hanya menjalankan saja dan selebihnya, terserah plug-in.

Berikut ini adalah source code 6.c:

```
#include <stdio.h>
#include <lua.h>
#include <lauxlib.h>

int main(void)
{
  lua_State *L;
  L = lua_open();
  luaL_openlibs(L);

  fprintf (stdout, [host] Selamat
datang di C\n);
  fprintf (stdout, [host] I/O
dilakukan oleh plugin\n);

  luaL_loadfile(L,6.lua);
  lua_call (L, 0, 0);
  lua_close (L);;
  fprintf (stdout, [host] Kembali
ke C\n);
  return 0;
};
```

Berikut ini adalah isi script 6.lua:

```
io.write ('Menghitung akar kuadrat\
n')
io.write ('Masukkan bilangan: ')
bil = io.stdin:read('*number')
io.write ('Anda memasukkan: ' .. bil
.. '\n')
result = math.sqrt (bil)
io.write ('Akar kuadrat dari ' ..
bil .. ' adalah ' .. result .. '\n')
```

Penjelasan Lua:

- io.stdin:read() dapat digunakan untuk membaca dari stdin.
- Argumen:

- *all: membaca semua file.
- *line: membaca baris berikut.
- *number: membaca bilangan.
- <num>: membaca string sampai <num> karakter.

Program 7: Logika oleh plug-in, I/O oleh host

Di program 7 ini, plugin (7.lua) hanya melakukan logika. Semua I/O tetap ditangani oleh host, yang diharapkan bisa menjaga konsistensi UI.

Berikut ini adalah source code 7.c:

```
#include <stdio.h>
#include <lua.h>
#include <lauxlib.h>

int main(int argc, char * argv[])
{
  float bil;
  float result;
  lua_State *L;
  L = lua_open();
  luaL_openlibs(L);
  fprintf (stdout, [host] Selamat
datang di C\n);
  fprintf (stdout, [host] I/O
dengan C, proses oleh plugin\n);
  fprintf (stdout, [host]
Menghitung akar kuadrat\n);
  luaL_loadfile(L,7.lua);
  lua_call (L,0,0);
  fprintf (stdout, [host] Masukkan
bilangan: );
  fscanf (stdin, %f, &bil);
  fprintf (stdout, [host] Anda
memasukkan: %f\n, bil);
  lua_getfield (L, LUA_GLOBALSINDEX,
akarkuadrat);
  lua_pushnumber (L, bil);
  lua_call (L, 1, 1);
  result = (float) lua_tonumber (L,
-1);
  lua_close (L);;
  fprintf (stdout, [host] Akar
kuadrat dari %f adalah %f\n,
bil,result);
  fprintf (stdout, [host] Kembali
ke C\n);
  return 0;
};
```

Berikut ini adalah isi script 7.lua:

```
function akarkuadrat (bil)
  return math.sqrt (bil)
end
```

Contoh output:

```
$ ./7
[host] Selamat datang di C
[host] I/O dengan C, proses oleh
plugin
[host] Menghitung akar kuadrat
[host] Masukkan bilangan: 10
[host] Anda memasukkan: 10.000000
[host] Akar kuadrat dari 10.000000
adalah 3.162278
[host] Kembali ke C
```

Penjelasan C:

- Bacalah juga penjelasan program 5.
- Untuk memanggil fungsi akar kuadrat:
  - Push nama fungsi.
    ```
    lua_getfield (L, LUA_
    GLOBALSINDEX, "akarkuadrat");
    ```
  - Push parameter.
    ```
    lua_pushnumber (L, bil);
    ```
  - Panggil dengan satu argumen dan satu return value.
    ```
    lua_call (L, 1, 1);
    ```
- Dapatkan hasil:
  ```
  result = (float) lua_tonumber (L,
  -1);
  ```

Penjelasan Lua:

untuk mendapatkan akar kuadrat, kita menggunakan math.sqrt().

Program 8: Plug-in umum, metadata dari plug-in

Semua contoh sebelumnya mengharuskan kita untuk mengetahui persis isi plug-in. Cobalah lihat contoh 7 sebelumnya. Kita telah mengetahui bahwa plug-in 7. lua akan menghitung akar kuadrat dan fungsi utama membutuhkan satu parameter. Kita telah mengetahuinya terlebih dahulu.

Dengan demikian, developer plug-in dan developer program utama haruslah berhubungan secara dekat, kalau tidak mau dikatakan sebagai tim yang sama. Pendek kata, kita hanya memindahkan sebagian tugas ke lua.

Di contoh-contoh sebelumnya, kita hanya membaca metadata informasi plug-in dari file plug-in, seperti nama, pembuat dan sebagainya. Kita belum membaca berapa argumen yang dibutuhkan, tipe argumen apa saja, dan tipe return value.

Di program 8 ini, kita akan mengusahakan untuk membangun program utama yang bisa menerima sembarang plug-in, selama plug-in tersebut menyediakan informasi-informasi berikut ini:

- plugin_name: nama plug-in.
- plugin_author: pembuat plug-in.
- plugin_copyright_string: string copyright plug-in.
- plugin_license: lisensi plug-in.
- plugin_return: return value fungsi utama, dengan nilai yang mungkin adalah "number" atau "string".
- plugin_argc: jumlah argumen yang bisa diterima fungsi utama.
- plugin_argv_<n>: tipe argumen ke <n>, dengan nilai yang mungkin adalah "number" atau "string".
- Sebagai catatan, setiap plugin memiliki satu fungsi utama dengan nama plugin_main().
- Kita perlu memastikan informasi-informasi tersebut sinkron, sebagai contoh Kalau plugin_argc=2, maka harus tersedia plugin_argv_1 dan plugin_argv_2.

Berikut adalah source code 8.c:

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

int main(int argc, char * argv[])
{
  char plugin_name[255];
  char field_name[255];
  char plugin_return[255];
  char temp[255];
  int plugin_argc;
  int i;
  char inputs[255];
  float inputf;
  char results[255];
  float resultf;

  /* -1=undefined, 0=number,
1=string*/
  int argv_type[255];
  lua_State *L;
  if (argc != 2)
  {
   fprintf (stderr, usage: %s
<plugin_file>\n, argv[0]);
   return 1;
  }
  L = lua_open();
  luaL_openlibs(L);
  fprintf (stdout, [host] Selamat
```

```
datang di C\n);
  fprintf (stdout, [host] Bekerja
dengan plugin: %s\n, argv[1]);

  luaL_loadfile(L,argv[1]);
  lua_call (L,0,0);

  lua_getfield (L,LUA_GLOBALSINDEX,
plugin_name);
  lua_getfield (L,LUA_GLOBALSINDEX,
plugin_return);
  lua_getfield (L,LUA_GLOBALSINDEX,
plugin_argc);

  strcpy (plugin_name, lua_
tostring(L,-3));
  strcpy (plugin_return, lua_
tostring(L,-2));
  plugin_argc = lua_tonumber(L,-1);

  fprintf (stdout, [host] Nama
plugin: %s\n, plugin_name);
  fprintf (stdout, [host] Jumlah
argumen: %d\n, plugin_argc);
  fprintf (stdout, [host] Return
type: %s\n, plugin_return);

  argv_type[0] = -1;

  for (i=1; i<=plugin_argc; i++)
  {
   sprintf (field_name, plugin_argv_
%d, i);
   lua_getfield (L,LUA_GLOBALSINDEX,
field_name);
   strcpy (temp, lua_tostring(L,-
1));

   if (strcmp (temp, number) == 0)
   {
     argv_type[i] = 0;
   }
   else if (strcmp (temp, string)
== 0)
   {
     argv_type[i] = 1;
   }

  }

  lua_getfield (L,LUA_GLOBALSINDEX,
plugin_main);

  for (i=1; i<=plugin_argc; i++)
  {
   if (argv_type[i] == 0)
   {
     strcpy(temp, number);
   }
   else if (argv_type[i] == 1)
   {
     strcpy(temp, string);
   };

   fprintf (stdout, \t[host]
Masukkan argumen ke %d [tipe: %s]:
, i,temp);

   if (strcmp (temp, number) == 0)
   {
     fscanf (stdin, %f, &inputf);
     fprintf (stdout, \t[host] Anda
memasukkan: %f \n, inputf);
     lua_pushnumber (L, inputf);
   }
   else if (strcmp (temp, string)
== 0)
   {
     fscanf (stdin, %s, inputs);
     fprintf (stdout, \t[host] Anda
memasukkan: %s \n, inputs);
     lua_pushstring (L, inputs);
   }
  }

  if (lua_pcall (L, plugin_argc,
1,0) != 0)
  {
   fprintf (stderr, Kesalahan
menjalankan plugin: %s\n, lua_
tostring(L, -1));
  }

  if (strcmp (plugin_
return,number) == 0)
  {
   resultf = lua_tonumber (L, -1);
   fprintf (stdout, [host] Return
value: %f \n, resultf);
  }
  else if (strcmp (plugin_return,
string) == 0)
  {
   strcpy(results, lua_tostring (L,
-1));
   fprintf (stdout, [host] Return
value: %s \n, results);
  }
  lua_close (L);;

  fprintf (stdout, [host] Kembali
ke C\n);

  return 0;
};
```

Contoh plugin 8-1.lua:

```
plugin_name = akarkuadrat
plugin_author = Noprianto
plugin_copyright_string = (c)
Noprianto, 2007
plugin_license = GPL

plugin_return = number
plugin_argc = 1
plugin_argv_1 = number

function plugin_main(n)
  return math.sqrt(n)
end
```

Penjelasan C:

- Dapatkan plugin_argc
- Perulangan pertama: kita mengulang sebanyak plugin_argc, dimulai dari 1, untuk mendapatkan tipe setiap argumen. Selanjutnya, semua tipe argumen kita simpan ke array argv_type dengan aturan: -1: tidak didefinisikan, 0: number, 1: string.
- Kita push nama fungsi.
- Perulangan kedua: kita mengulang sebanyak plugin_argc, dimulai dari 1, dan meminta input user dengan menyebutkan tipe data yang diharapkan. Untuk setiap input, kita push ke stack.
- Kita memanggil lua_call dengan plugin_argc argumen dan 1 return value
- Kita, yang telah mengetahui tipe kembalian fungsi utama, kemudian mendapatkan kembalian dari fungsi utama

Dengan demikian, program utama tidak harus tahu persis isi plug-in. Plug-in bisa dikembangkan siapa saja, selama menaati aturan plug-in. Anda, tentu bisa membuat aturan yang lebih baik dan lebih disukai. Yang kita bahas di sini adalah salah satu contoh.

Plug-in adalah topik yang kompleks. Di pembahasan ini, hanya sedikit yang kita bahas. Di lain kesempatan, kita akan membahas berbagai contoh lain. Selamat mencoba!■

Noprianto [noprianto@infolinux.co.id]

IKLAN

Otomatis Membaca dan Menulis Media Removeable

Ingin langsung melakukan tindakan tertentu pada saat media *removeable* ditancapkan ke komputer? Ingin membaca semua data, melakukan pemformatan, atau meng-*copy*-kan data tertentu? Dengan bantuan *shell script*, kita bisa melakukannya dengan cepat dan mudah.

Tujuan dari tulisan ini tentu bukan untuk pencurian data, misalnya ketika CDROM atau USB flash disk dimasukkan, maka diam-diam data akan di-*copy*-kan. Akan terasa lebih berguna kalau Anda membangun semacam *public station* yang akan melakukan tindakan tertentu begitu *removeable device user* ditancapkan. Misalnya, pada media promosi program komputer. Begitu user memasukkan USB flash disk, maka program komputer tersebut bisa langsung di-copy-kan tanpa user harus repot-repot meng-copy-nya sendiri. Tentu saja, aksi yang ingin dilakukan bisa sangat beragam. Meng-copy, menulis, atau memformat hanyalah sebagian dari contoh yang bisa dilakukan.

Aksi otomatis ini tentu saja bisa dilakukan kalau kita memonitor device tertentu. Di Linux, perihal device monitoring ini cukup beragam. Di tulisan ini, kita akan menggunakan cara yang cukup tradisional, namun bisa berjalan di sebagian besar distro Linux. Di kesempatan-kesempatan berikutnya, kita akan membahas cara yang lebih modern.

Semua contoh yang ada di tulisan ini dibangun dengan shell script dan memanfaatkan program dialog untuk membangun *user interface* yang lebih baik. Oleh karenanya, pastikanlah dialog telah terinstal di sistem Anda, atau modifikasilah script agar tidak menggunakan dialog.

Beberapa contoh yang ada di tulisan ini (untuk media USB Flash Disk) mengharapkan Anda menggunakan Linux 2.6 ke atas yang mengaktifkan udev. Hampir semua distro modern sudah menggunakan linux 2.6 dan mengaktifkan udev. Bagi Anda yang memiliki alasan untuk tidak menggunakan udev atau distro/kernel yang digunakan tidak mendukung, beberapa modifikasi bisa dilakukan agar script tetap dapat bekerja.

Tulisan ini dibangun di atas sistem slackware 11, kernel 2.6.18, namun seharusnya bisa diterapkan pada berbagai sistem lain tanpa masalah.

Floppy disk

Pertama-tama, kita akan membahas tentang otomatis membaca dan menulis dari dan ke floppy disk, dengan opsi untuk pembuatan dan penulisan *image*. Untuk mendeteksi ada atau tidaknya floppy disk, kita menggunakan cara yang tradisional, yaitu dengan mencoba melakukan mount dan apabila berhasil, maka floppy kita anggap dimasukkan dan valid dan oleh karena itu, bisa diproses lebih lanjut.

Kenapa kita juga membuat image dari floppy atau menulis image ke floppy? Seperti kita ketahui bersama, terkadang kita ingin mengopikan satu *rescue system* atau distro mini ke floppy. Tentu saja, tidak semua sistem bisa di-copy-kan satu file demi satu file. Untuk kondisi seperti ini, lebih baik kita menggunakan image floppy. Image floppy adalah sebuah file yang isinya sama persis dengan isi floppy dari ujung ke ujung. Penulisan atau pembacaannya dilakukan menggunakan program dd.

watch_floppy_disk_read.sh

Sebagai contoh pertama, kita akan membaca secara otomatis dari floppy. Pembacaan dilakukan file per file dan juga diikuti dengan pembuatan image floppy.

Berikut ini adalah source code watch_floppy_disk_read.sh:

```
#!/bin/sh

#nop, 2007, GPL

DEV=/dev/fd0
LOGFILE=/tmp/`basename $0`.log
DD_BS=100k

alias ADIALOG=dialog --backtitle
'AutoRead (FLOPPY DISK)'
DIALOG_SIZE=10 40

if [ $(id -u) -ne 0 ]
then
  ADIALOG --msgbox 'Jalankanlah
aplikasi ini sebagai root' $DIALOG_
SIZE
  exit 1
fi

while [ 1 ]
do
  DESTDIR=/tmp/floppy-data/`date +%d-
%m-%Y--%H:%M:%S`
  DESTDIR_CONTENT=$DESTDIR/content
  DESTDIR_IMAGE=$DESTDIR/floppy.img
```

```
  TEMP=/tmp/$0.`date +%s`.temp

  mkdir -p $TEMP 1>/dev/null 2>&1

  mount $DEV $TEMP 1>/dev/null 2>&1
  if [ $? -ne 0 ]
  then
   ADIALOG --infobox 'IDLE: Masukkan
Floppy Disk Anda' $DIALOG_SIZE
  else
   ADIALOG --infobox 'Floppy Disk
terdeteksi' $DIALOG_SIZE
   sleep 1

   #copying data
   ADIALOG --infobox 'Tunggulah
sebentar...\nSedang membaca data'
$DIALOG_SIZE
   [ -d $DESTDIR_CONTENT ] && rm -rf
$DESTDIR_CONTENT 1>/dev/null 2>&1
   mkdir -p $DESTDIR 2>>$LOGFILE
   cp -af $TEMP/ $DESTDIR_CONTENT
2>>$LOGFILE
   umount $TEMP 2>> $LOGFILE
   #end

   #creating image
   ADIALOG --infobox 'Tunggulah
sebentar...\nSedang membuat image'
$DIALOG_SIZE
   dd if=$DEV of=$DESTDIR_IMAGE
bs=$DD_BS seek=0 1>/dev/null 2>&1
   [ $? -ne 0 ] && echo dd-error
>> $LOGFILE
   #end

   TEST=`wc -c $LOGFILE | cut -d' '
-f1`

   if [ $TEST -ne 0 ]
   then
    ADIALOG --msgbox 'Terjadi
kesalahan dalam pembacaan.\
nKeluarkanlah Floppy Disk Anda,\
nKemudian tekan ENTER' $DIALOG_SIZE
   else
    ADIALOG --msgbox 'Selesai.\
nKeluarkanlah Floppy Disk Anda,\
nKemudian tekan ENTER' $DIALOG_SIZE
   fi

   rm -f $LOGFILE

  fi
  sleep 1
  rmdir $TEMP
done

unalias ADIALOG
rmdir $TEMP
```

Berikanlah hak akses executable dengan perintah berikut:

```
$ chmod +x watch_floppy_disk_read.sh
```

Penjelasan:

- Pada saat dijalankan, program akan memeriksa apakah dijalankan oleh root atau bukan. Selain root, untuk sederhananya, kita menolak melanjutkan.
- Program akan mengulang terus menerus:
  - Pertama-tama, kita menentukan direktori tujuan dengan pola /tmp/floppy-data/<dd-mm-yyyy—hh:mm:ss>. Contoh: /tmp/floppy-data/17-09-2007—19:44:56. Direktori tujuan tersebut akan berisikan hasil pengopian sesuai dengan tanggal dan jam. Di dalam direktori tersebut, akan terdapat direktori content (yang berisikan file-file yang di-copy-kan dari floppy) dan sebuah file floppy.img yang merupakan image floppy.
  - Kita membuat pula sebuah direktori (variabel TEMP) dengan memasukkan unsur waktu ke dalam nama. Direktori tersebut akan digunakan sebagai mount point device floppy.
  - Kita akan coba melakukan mount floppy (variabel DEV) ke TEMP.
  - Apabila gagal mount, maka pesan IDLE akan ditampilkan.
  - Apabila berhasil mount:
    - Kita meng-copy data file demi file. Pesan kesalahan disimpan di LOGFILE.
    - Unmount floppy.
    - Kita membuat image floppy. Pesan kesalahan akan disimpan di LOGFILE
    - Kita memeriksa ukuran LOGFILE. Apabila lebih dari 0 byte, maka terjadi kesalahan. Selanjutnya, status pembacaan akan kita tampilkan.
    - User akan mengeluarkan floppy dari drive-nya dan menekan tombol ENTER.
  - Tunda 1 detik dan hapus TEMP.
- Terakhir, kita menghapus TEMP.
- Ubahlah device yang ingin dimonitor di variable DEV.
- Sesuaikan *block size* pembacaan image (program dd) di variabel DD_BS. Bacalah manual dd untuk informasi selengkapnya.
- Sesuaikan script dengan kebutuhan Anda. Proses peng-copy-an yang dilakukan di sini dilakukan dua kali: peng-copy-an file-per-file dan pembuatan image.

watch_floppy_disk_write.sh

Di contoh ini, kita akan langsung menulis ke floppy begitu floppy yang mount-able dimasukkan. Penulisan dilakukan file-per-file, dengan file-file sumber disimpan di /tmp/floppy-src/content/.

Berikut ini adalah source code watch_floppy_disk_write.sh:

```
#!/bin/sh

#nop, 2007, GPL

DEV=/dev/fd0
LOGFILE=/tmp/`basename $0`.log
SRCDIR_CONTENT=/tmp/floppy-src/
content

alias ADIALOG=dialog --backtitle
'AutoWrite (FLOPPY DISK - CONTENT)'
DIALOG_SIZE=10 40

if [ $(id -u) -ne 0 ]
then
 ADIALOG --msgbox 'Jalankanlah
aplikasi ini sebagai root' $DIALOG_
SIZE
 exit 1
fi



while [ 1 ]
do
 TEMP=/tmp/$0.`date +%s`.temp

 mkdir -p $TEMP 1>/dev/null 2>&1

 mount $DEV $TEMP 1>/dev/null 2>&1
 if [ $? -ne 0 ]
 then
  ADIALOG --infobox 'IDLE: Masukkan
Floppy Disk Anda' $DIALOG_SIZE
 else
  ADIALOG --infobox 'Floppy Disk
terdeteksi' $DIALOG_SIZE
```

```
   sleep 1

   #copying data to floppy
   ADIALOG --infobox 'Tunggulah
sebentar...\nSedang mengopi data ke
floppy' $DIALOG_SIZE
   cp -af $SRCDIR_CONTENT $TEMP
2>>$LOGFILE
   umount $TEMP 2>> $LOGFILE
   #end

   TEST=`wc -c $LOGFILE | cut -d' '
-f1`

   if [ $TEST -ne 0 ]
   then
     ADIALOG --msgbox 'Terjadi
kesalahan dalam penulisan.\
nKeluarkanlah Floppy Disk Anda,\
nKemudian tekan ENTER' $DIALOG_SIZE
   else
     ADIALOG --msgbox 'Selesai.\
nKeluarkanlah Floppy Disk Anda,\
nKemudian tekan ENTER' $DIALOG_SIZE
   fi

   rm -f $LOGFILE

  fi
  sleep 1
  rmdir $TEMP
done

unalias ADIALOG
rmdir $TEMP
```

Berikanlah hak akses executable dengan perintah berikut:

```
$ chmod +x watch_floppy_disk_write.sh
```

Penjelasan:

- Bacalah juga pembahasan tentang watch_floppy_disk_read.sh.
- Pembacaan dilakukan dengan program cp dengan stderr disimpan pada LOGFILE.
- Sesuaikan variabel SRCDIR_CONTENT dengan sistem Anda.

watch_floppy_disk_write_img.sh

Setelah menulis file-per-file, di contoh ini, kita akan menulis ke floppy dengan sumber berupa image floppy yang telah disiapkan terlebih dahulu. Untuk membuat image, lihatlah source code watch_floppy_disk_read.sh.

Berikut ini adalah source code watch_floppy_disk_write_img.sh:

```
#!/bin/sh

#nop, 2007, GPL

DEV=/dev/fd0
LOGFILE=/tmp/`basename $0`.log
SRCDIR_IMAGE=/tmp/floppy-src/floppy.
img
DD_BS=100k

alias ADIALOG=dialog --backtitle
'AutoWrite (FLOPPY DISK - IMAGE)'
DIALOG_SIZE=10 40

if [ $(id -u) -ne 0 ]
then
  ADIALOG --msgbox 'Jalankanlah
aplikasi ini sebagai root' $DIALOG_
SIZE
  exit 1
fi




while [ 1 ]
do
  TEMP=/tmp/$0.`date +%s`.temp

  mkdir -p $TEMP 1>/dev/null 2>&1

  mount $DEV $TEMP 1>/dev/null 2>&1
  if [ $? -ne 0 ]
  then
   ADIALOG --infobox 'IDLE: Masukkan
Floppy Disk Anda' $DIALOG_SIZE
  else
   ADIALOG --infobox 'Floppy Disk
terdeteksi' $DIALOG_SIZE
   sleep 1
   umount $DEV

   #creating image
   ADIALOG --infobox 'Tunggulah
sebentar...\nSedang menulis image ke
floppy' $DIALOG_SIZE
   dd if=$SRCDIR_IMAGE of=$DEV
bs=$DD_BS seek=0 1>/dev/null 2>&1
   [ $? -ne 0 ] && echo dd-error
>> $LOGFILE
   end

   TEST=`wc -c $LOGFILE | cut -d' '
-f1`

   if [ $TEST -ne 0 ]
   then
     ADIALOG --msgbox 'Terjadi
kesalahan dalam penulisan.\
nKeluarkanlah Floppy Disk Anda,\
nKemudian tekan ENTER' $DIALOG_SIZE
   else
     ADIALOG --msgbox 'Selesai.\
nKeluarkanlah Floppy Disk Anda,\
nKemudian tekan ENTER' $DIALOG_SIZE
   fi

   rm -f $LOGFILE

  fi
  sleep 1
  rmdir $TEMP
done

unalias ADIALOG
rmdir $TEMP
```

Berikanlah hak akses executable dengan perintah berikut:

```
$ chmod +x watch_floppy_disk_write_
img.sh
```

Penjelasan:

- Bacalah juga pembahasan tentang watch_floppy_disk_read.sh.
- Sesuaikan variabel SRCDIR_IMAGE (lokasi image floppy) dengan sistem Anda.

USB Flash Disk

Berikut ini, kita akan membahas pembacaan dan penulisan otomatis pada USB Flash Disk. Untuk saat ini, pendeteksian dilakukan menggunakan bantuan udev, di mana *device node* akan *dicreate* otomatis begitu device yang bersangkutan dimasukkan. Hanya, cara yang digunakan di tulisan ini sangat sederhana, karena hanya memonitor sebuah device yang sudah diset terlebih dahulu, misal /dev/sda1.

Bagi Anda yang ingin menyesuaikan dengan device sesungguhnya, tanpa harus mengunci di /dev/sda1, bisa juga dengan membaca output dmesg atau memanfaatkan output lshal.

Di kesempatan lain, kita akan membahas udev dan pendeteksian *hardware* lebih lanjut.

Untuk USB flash disk, kita tidak membuat image, melainkan hanya pengcopy-an file-per-file.

IKLAN

watch_usb_flash_disk_read.sh

Berikut ini adalah source code watch_usb_flash_disk_read.sh. Pembahasan dapat dibaca setelah source code:

```
#!/bin/sh

#nop, 2007, GPL

DEV=/dev/sda1
LOGFILE=/tmp/`basename $0`.log

alias ADIALOG=dialog --backtitle 'AutoRead (USB FLASH DISK)'
DIALOG_SIZE=10 40

if [ $(id -u) -ne 0 ]
then
  ADIALOG --msgbox 'Jalankanlah aplikasi ini sebagai root' $DIALOG_SIZE
  exit 1
fi



while [ 1 ]
do
  DESTDIR=/tmp/usbflashdisk-data/`date +%d-%m-%Y--%H:%M:%S`
  DESTDIR_CONTENT=$DESTDIR/content

  if [ ! -e $DEV ]
  then
   ADIALOG --infobox 'IDLE: Masukkan USB Flash Disk Anda' $DIALOG_SIZE
  else
   ADIALOG --infobox 'USB Flash Disk terdeteksi' $DIALOG_SIZE
   sleep 1

   TEMP=/tmp/$0.`date +%s`.temp

   mkdir -p $TEMP 1>/dev/null 2>&1

   mount $DEV $TEMP 1>/dev/null 2>&1


   #copying data
   ADIALOG --infobox 'Tunggulah sebentar...\nSedang membaca data' $DIALOG_SIZE
   [ -d $DESTDIR_CONTENT ] && rm -rf $DESTDIR_CONTENT 1>/dev/null 2>&1
   mkdir -p $DESTDIR 2>>$LOGFILE
   cp -af $TEMP/ $DESTDIR_CONTENT 2>>$LOGFILE
   umount $TEMP 2>> $LOGFILE
   #end

   TEST=`wc -c $LOGFILE | cut -d' ' -f1`

   if [ $TEST -ne 0 ]
   then
    ADIALOG --msgbox 'Terjadi kesalahan dalam pembacaan.\nKeluarkanlah USB Flash Disk Anda,\nKemudian tekan ENTER' $DIALOG_SIZE
   else
    ADIALOG --msgbox 'Selesai.\nKeluarkanlah USB Flash Disk Anda,\nKemudian tekan ENTER' $DIALOG_SIZE
   fi

   rm -f $LOGFILE

   rmdir $TEMP
  fi
  sleep 1
done

unalias ADIALOG
rmdir $TEMP
```

Berikanlah hak akses executable dengan perintah berikut:

```
$ chmod +x watch_usb_flash_disk_read.sh
```

Penjelasan:

- Bacalah juga pembahasan tentang watch_floppy_disk_read.sh.
- Semua file-file yang dibaca akan disimpan di /tmp/usbflashdisk-data/<dd-mm-yyyy—hh:mm:ss>/content/. Ubahlah variable DESTDIR dan DESTDIR_CONTENT apabila diperlukan.
- Untuk pendeteksian device, kita hanya memeriksa keberadaan DEV (/dev/sda1). Apabila ditemukan, maka kita akan melanjutkan proses.
- Mount dilakukan pada TEMP setelah device terdeteksi.
- Pengopian dilakukan dengan program cp dengan stderr diredireksi ke LOGFILE.
- Setelah peng-copy-an, umount segera dilakukan.

watch_usb_flash_disk_write.sh

Berikut ini adalah source code watch_usb_flash_disk_write.sh. Pembahasan dapat dibaca setelah source code:

```
#!/bin/sh

#nop, 2007, GPL

DEV=/dev/sda1
LOGFILE=/tmp/`basename $0`.log
SRCDIR_CONTENT=/tmp/usbflashdisk-src/content

alias ADIALOG=dialog --backtitle 'AutoWrite (USB FLASH DISK)'
DIALOG_SIZE=10 40

if [ $(id -u) -ne 0 ]
then
  ADIALOG --msgbox 'Jalankanlah aplikasi ini sebagai root' $DIALOG_SIZE
  exit 1
fi



while [ 1 ]
do
  if [ ! -e $DEV ]
  then
   ADIALOG --infobox 'IDLE: Masukkan USB Flash Disk Anda' $DIALOG_SIZE
  else
   ADIALOG --infobox 'USB Flash Disk terdeteksi' $DIALOG_SIZE
   sleep 2

   TEMP=/tmp/$0.`date +%s`.temp

   mkdir -p $TEMP 1>/dev/null 2>&1

   mount $DEV $TEMP 1>/dev/null 2>&1

   #copying data to USB Flash Disk
   ADIALOG --infobox 'Tunggulah sebentar...\nSedang mengopi data ke USB Flash Disk' $DIALOG_SIZE
   cp -af $SRCDIR_CONTENT $TEMP 2>>$LOGFILE
   umount $TEMP 2>> $LOGFILE
   #end

   TEST=`wc -c $LOGFILE | cut -d' ' -f1`

   if [ $TEST -ne 0 ]
   then
```

```
    ADIALOG --msgbox 'Terjadi
kesalahan dalam penulisan.\
nKeluarkanlah USB Flash Disk Anda,\
nKemudian tekan ENTER' $DIALOG_SIZE
  else
    ADIALOG --msgbox 'Selesai.\
nKeluarkanlah USB Flash Disk Anda,\
nKemudian tekan ENTER' $DIALOG_SIZE
  fi

  rm -f $LOGFILE
  rmdir $TEMP

 fi
 sleep 1
done

unalias ADIALOG
rmdir $TEMP
```

Berikanlah hak akses executable dengan perintah berikut:

```
$ chmod +x watch_usb_flash_disk_
write.sh
```

Penjelasan:

- Bacalah juga pembahasan tentang watch_floppy_disk_read.sh dan watch_usb_flash_disk_read.sh.
- File-file sumber disimpan di /tmp/usb-flashdisk-src/content. Ubahlah variable SRCDIR_CONTENT apabila diperlukan.

CDROM

Untuk pembacaan CDROM, kita menggunakan cara yang mirip dengan floppy, dimana kita akan mencoba melakukan mount dan apabila berhasil dilakukan, barulah peng-copy-an dan pembuatan image dilakukan.

Sementara, untuk penulisan/pembakaran, cara yang dilakukan sedikit berbeda. Penulis di sini mengasumsikan di mana media yang dimasukkan adalah CDR. Kita tidak akan memproses CDRW yang masih berisikan data (sehingga harus di-blank terlebih dahulu). Pembakaran juga dilakukan tanpa multi-session.

Bagaimana mendeteksi bahwa CDR dimasukkan, sebelum kita membakar? Cara yang digunakan saat ini masih sangat tradisional, yaitu dengan menjalankan program cdrecord dan menampilkan data ATIP (Absolute Time In Pre-groove). Cdrecord akan keluar dengan exit code nonsukses apabila media tidak valid. Untuk saat ini, kita hanya memeriksa exit code tanpa membaca informasi ATIP lebih detil.

Berikut ini adalah contoh informasi ATIP untuk media CDR:

```
# cdrecord dev=1,0,0 -atip
...
...
ATIP info from disk:
  Indicated writing power: 5
  Is not unrestricted
  Is not erasable
  Disk sub type: Medium Type B, low
Beta category (B-) (4)
  ATIP start of lead in:  -11811
(97:24/39)
  ATIP start of lead out: 359849
(79:59/74)
Disk type:    Short strategy type
(Phthalocyanine or similar)
Manuf. index: 41
Manufacturer: UNITECH JAPAN INC.
```

Exit code yang kita dapatkan adalah:

```
# echo $?
0
```

Apabila media tidak dimasukkan, output yang didapat di sistem penulis adalah:

```
# cdrecord dev=1,0,0 -atip
...
...
Sense flags: Blk 0 (not valid)
cmd finished after 0.011s timeout 40s
cdrecord: No disk / Wrong disk!
```

Exit code yang kita dapatkan adalah:

```
# echo $?
255
```

Tentu saja, kita harus mengetahui device CD Writer kita, yang bisa didapatkan dengan perintah berikut:

```
# cdrecord -scanbus
...
...
...
Using libscg version 'schily-0.8'.
scsibus1:
        1,0,0   100) 'LITE-ON '
'LTR-52246S      ' '6S0D' Removable
CD-ROM
        1,1,0   101) *
        1,2,0   102) *
        1,3,0   103) *
...
...
,,,
```

watch_cdrom_read.sh

Berikut ini adalah source code watch_cdrom_read.sh. Pembahasan dilakukan setelah source code:

```
#!/bin/sh

#nop, 2007, GPL

DEV=/dev/cdrom
LOGFILE=/tmp/`basename $0`.log
DD_BS=10M

alias ADIALOG=dialog --backtitle
'AutoRead (CDROM)'
DIALOG_SIZE=10 40

if [ $(id -u) -ne 0 ]
then
  ADIALOG --msgbox 'Jalankanlah
aplikasi ini sebagai root' $DIALOG_
SIZE
  exit 1
fi




while [ 1 ]
do
  DESTDIR=/tmp/cdrom-data/`date +%d-
%m-%Y--%H:%M:%S`
  DESTDIR_CONTENT=$DESTDIR/content
  DESTDIR_IMAGE=$DESTDIR/cdrom.iso

  TEMP=/tmp/$0.`date +%s`.temp

  mkdir -p $TEMP 1>/dev/null 2>&1

  mount $DEV $TEMP 1>/dev/null 2>&1
  if [ $? -ne 0 ]
  then
   ADIALOG --infobox 'IDLE: Masukkan
CDROM Anda' $DIALOG_SIZE
  else
   ADIALOG --infobox 'CDROM
terdeteksi' $DIALOG_SIZE
   sleep 1

   #copying data
   ADIALOG --infobox 'Tunggulah
sebentar...\nSedang membaca data'
$DIALOG_SIZE
   [ -d $DESTDIR_CONTENT ] && rm -rf
$DESTDIR_CONTENT 1>/dev/null 2>&1
   mkdir -p $DESTDIR 2>>$LOGFILE
   cp -af $TEMP/ $DESTDIR_CONTENT
```

```
2>>$LOGFILE
   umount $TEMP 2>> $LOGFILE
   #end

   #creating image
   ADIALOG --infobox 'Tunggulah
sebentar...\nSedang membuat image'
$DIALOG_SIZE
   dd if=$DEV of=$DESTDIR_IMAGE
bs=$DD_BS seek=0 1>/dev/null 2>&1
   [ $? -ne 0 ] && echo "dd-error"
>> $LOGFILE
   #end

   TEST=`wc -c $LOGFILE | cut -d' '
-f1`

   if [ $TEST -ne 0 ]
   then
     ADIALOG --msgbox 'Terjadi
kesalahan dalam pembacaan.\
nKeluarkanlah CDROM Anda,\nKemudian
tekan ENTER' $DIALOG_SIZE
   else
     ADIALOG --msgbox 'Selesai.\
nKeluarkanlah CDROM Anda,\nKemudian
tekan ENTER' $DIALOG_SIZE
   fi

   rm -f $LOGFILE

  fi
  sleep 1
  rmdir $TEMP
done

unalias ADIALOG
rmdir $TEMP
```

Berikanlah hak akses executable dengan perintah berikut:

```
$ chmod +x watch_cdrom_read.sh
```

Penjelasan:

- Bacalah juga pembahasan tentang watch_floppy_disk_read.sh.
- Untuk DD_BS, kita bisa memberikan nilai yang cukup besar, mempertimbangkan ukuran CDROM.

watch_cdrom_write.sh

Sebelum melakukan penulisan, pastikan image tersedia terlebih dahulu. Untuk membuat image dari CDROM, lihatlah contoh sebelumnya. Untuk membuat image dari direktori yang berisikan data, gunakanlah program mkisofs. Bacalah manual mkisofs untuk informasi selengkapnya.

Berikut ini adalah source code watch_cdrom_write.sh:

```
#!/bin/sh

#nop, 2007, GPL

DEV=0,0,0
DEV_NODE=/dev/sr0
SPEED=24
LOGFILE=/tmp/`basename $0`.log
SRCDIR_IMAGE=/tmp/cdrom-src/cdrom.
iso

alias ADIALOG=dialog --backtitle
'AutoWrite (CDROM)'
DIALOG_SIZE=10 40

if [ $(id -u) -ne 0 ]
then
  ADIALOG --msgbox 'Jalankanlah
aplikasi ini sebagai root' $DIALOG_
SIZE
  exit 1
fi



while [ 1 ]
do
  cdrecord -dev=$DEV -atip 1>/dev/
null 2>&1
  if [ $? -ne 0 ]
  then
   ADIALOG --infobox 'IDLE: Masukkan
CDROM Blank Anda' $DIALOG_SIZE
  else
   ADIALOG --infobox 'CDROM
terdeteksi' $DIALOG_SIZE
   sleep 1

   #creating image
   ADIALOG --infobox 'Tunggulah
sebentar...\nSedang membakar CD'
$DIALOG_SIZE
   cdrecord gracetime=2 dev=$DEV
speed=$SPEED $SRCDIR_IMAGE 1>/dev/
null 2>&1
   [ $? -ne 0 ] && echo cdrecord-
error >> $LOGFILE
   #end

   eject $DEV_NODE

   TEST=`wc -c $LOGFILE | cut -d' '
-f1`

   if [ $TEST -ne 0 ]
   then
     ADIALOG --msgbox 'Terjadi
kesalahan dalam proses burn.\
nTutup tray CDROM dan tekan ENTER.'
$DIALOG_SIZE
   else
     ADIALOG --msgbox 'Selesai.\
nTutup tray CDROM dan tekan ENTER.'
$DIALOG_SIZE
   fi

   rm -f $LOGFILE

  fi
  sleep 3
done

unalias ADIALOG
rmdir $TEMP
```

Berikanlah hak akses executable dengan perintah berikut:

```
$ chmod +x watch_cdrom_write.sh
```

Penjelasan:

- Bacalah juga pembahasan tentang watch_floppy_disk_read.sh
- DEV adalah device yang dikenal oleh cdrecord. Untuk eject, kita mempergunakan device node yang disimpan pada DEV_NODE.
- Aturlah kecepatan bakar di variable SPEED.
- Image CDROM disimpan di /tmp/cdrom-src/cdrom.iso.
- Perintah yang digunakan untuk burn:

```
cdrecord gracetime=2 dev=$DEV
speed=$SPEED $SRCDIR_IMAGE 1>/
dev/null 2>&1
```

- Untuk mencoba-coba (tanpa mengaktifkan laser), berikanlah opsi -dummy
- Setelah pembakaran dilakukan, kita akan mengeject device.

Sampai di sini dulu pembahasan kita. Tambahkan aksi yang Anda inginkan (auto make file system, auto blank CDRW, dan lainnya). Apabila diperlukan, periksa juga tipe file sistem (bisa dengan bantuan mount) dan ruang kosong (dengan df). Selamat mencoba dan mengembangkan.■

Noprianto [noprianto@infolinux.co.id]

IKLAN

Kontrol Akses File Menggunakan ACL

Pada *platform* Linux, kita mengenal adanya hak akses ke suatu file maupun direktori. Jika pengelolaan hak akses menjadi semakin kompleks, penggunaan *Access Control List* (ACL) menjadi tidak terelakkan. Dalam "Tutorial" kali ini, akan dijelaskan cara mengontrol hak akses suatu file maupun direktori dengan menggunakan ACL.

Sebagaimana kita ketahui bahwa Linux seperti Unix memiliki standar hak akses file baca, tulis dan eksekusi pada user, group, dan others. Standar hak akses file tersebut sudah mencukupi untuk file/directory yang hanya diakses oleh satu user dan satu group tertentu. Tetapi, terkadang kita dihadapkan pada masalah jika suatu file/directory harus dapat diakses oleh beberapa user dan beberapa group tertentu. Tentu saja kita masih dapat menggunakan hak akses ketiga, yaitu "others" untuk mengatasi hal tersebut. Namun hal tersebut akan mengakibatkan akses tak terbatas bagi semua user dan group yang tentunya bisa membahayakan keamanan sistem.

Penggunaan *Access Control List* (ACL) menjadi tak terelakkan manakala kita membutuhkan manajemen file dan directory yang lebih kompleks. ACL merupakan daftar hak akses tambahan untuk hak akses standard pada unix. ACL bagi user dan administrator akan memberikan fleksibilitas dan kontrol yang lebih baik terhadap hak akses pada file dan directory.

Sebagai contoh kasus kita akan coba untuk membuat beberapa user dan group serta directory dengan kondisi sebagai berikut:

- Terdapat 8 user (user1, user2, .., user8).
- Terdapat 9 group terdiri dari:
  - admin.
  - LC.
  - manager.
  - molding.
  - assembly.
  - stamping.
  - LC-assy.
  - LC-mold.
  - LC-stamp.
- Terdapat 6 directory yang terdiri dari:
  - DATA.
  - LC.
  - molding.
  - assembly.
  - stamping.
  - manager.

Pembuatan pengguna

Buatlah 8 pengguna dan set kata sandi untuk masing–masing pengguna. Nama dan jumlah pengguna dapat disesuaikan dengan kebutuhan.

```
# /usr/sbin/adduser user1
# /usr/bin/passwd user1
# /usr/sbin/adduser user2
# /usr/bin/passwd user2
```

Demikian seterusnya hingga terdapat 8 pengguna seperti pada contoh.

Pembuatan grup

Buatlah 9 grup dengan nama seperti pada contoh diatas. Pada kasus ini setiap pengguna akan dikelompokkan dalam grup–grup yang akan dibuat. Setiap grup minimal akan berisi satu anggota kelompok. Selanjutnya jika diinginkan dapat membuat pengguna lagi dan langsung dimasukkan ke grup yang bersangkutan sesuai kebutuhan. Hak akses yang akan diberikan pada directory akan berdasarkan grup dengan pertimbangan fleksibilitas. Mengingat adanya kemungkinan mutasi pengguna antargrup dan penghapusan akun pengguna.

```
# /usr/sbin/groupadd admin
# /usr/sbin/groupadd LC
```

Demikian seterusnya hingga terdapat 9 grup seperti pada contoh.

Pembuatan Directory

Selanjutnya kita akan buat directory–directory sesuai grup yang sudah ada sebelumnya. Dan setelah itu berikan hak akses pada directory "DATA" dan sub-directory di dalamnya.

```
# mkdir -p /DATA {LC,molding,assembl
y,manager,stamping}
# chmod -R 770 /DATA
# chmod 775 /DATA
```

Konfigurasi group owner pada directory

Tidak ada perubahan pada *setting* kepemilikan pada directory "DATA" dan untuk semua directory DATA dan sub directory di dalamnya kepemilikan tetap pada user "root". Sedangkan, kepemilikan group disesuaikan dengan nama dari directory itu.

```
# chgrp logistic /DATA/LC
# chgrp molding /DATA/molding
# chgrp assembly /DATA/assembly
# chgrp manager /DATA/manager
# chgrp stamping /DATA/stamping
```

Konfigurasi dukungan ACL pada filesystem

Umumnya pada tiap distribusi Linux secara standar filesystem tidak di-*mount* dengan dukungan ACL. Karena itulah pada filesystem tempat directory dibuat harus di-mount/*remount* dengan dukungan ACL. Beberapa hal yang perlu diperhatikan sebelum memulai dukungan ACL:

- Pada kernel 2.6 mendukung penggunaan ACL pada filesystem ext2, ext3, xfs, jfs, dan ReiserFS.
- Petunjuk pada artikel ini menggunakan distribusi Fedora Core 7 dengan filesystem ext3.
- Sebelum kita mulai pastikan bahwa kita akan memberikan konfigurasi ACL pada partisi selain /, /boot dan swap.
- Pastikan Anda memiliki hak akses sebagai root.

Setelah kondisi tersebut sudah terpenuhi, maka selanjutnya bisa kita mulai melakukan konfigurasi:

- Back-up file /etc/fstab.

```
# cp /etc/fstab /etc/fstab.bak
```

- Buka file /etc/fstab lalu pilih partisi yang akan diberikan dukungan ACL. Selanjutnya ubahlah opsi parameter mounting yang sebelumya bernilai "defaults" menjadi "rw,acl".

```
/dev/hdd1               /DATA
ext3          rw,acl              0
0
```

- Setelah modifikasi /etc/fstab dilakukan maka selanjutnya adalah mounting ulang filesystem /dev/hdd1.

```
# mount -v -o remount /DATA
```

- Setelah proses remount selesai dan berhasil maka akan ada pesan seperti dibawah yang menyatakan bahwa device /dev/hdd1 dengan mount point /DATA sudah di mounting ulang dengan format ext3 dan akses read-write serta support ACL.

```
/dev/hdd1 on /DATA ext3 (rw,acl)
```

Menambahkan ACL pada directory dan file

Untuk mengkonfigurasi ACL pada tiap directory kita akan gunakan perintah "setfacl" dengan tambahan parameter sesuai dengan kebutuhan. Parameter yang digunakan antara lain:

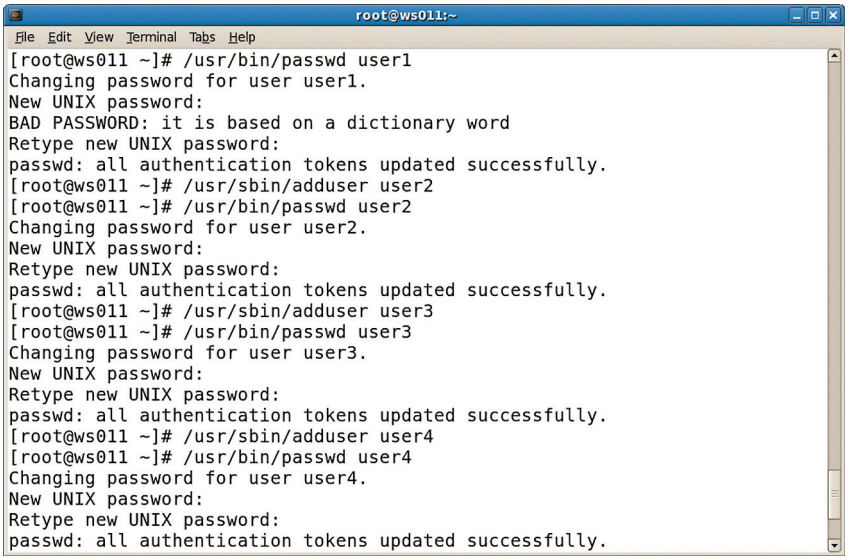
- -R : digunakan agar konfigurasi berlaku untuk directory terkait beserta directory yang berada didalamnya (rekursif).
- -d : digunakan untuk konfigurasi standar yang selanjutnya akan diberlakukan pada pembuatan file dan directory baru.
- -m : digunakan untuk memodifikasi konfigurasi standar ACL pada directory yang terkait.
- -b : digunakan untuk menghapus semua ACL tambahan

Berikut ini adalah konfigurasi acl yang dilakukan pada tiap – tiap directory:

```
# setfacl -R -m g:admin:rwx,g:
manager:rx ../DATA/
# setfacl -R -d -m g:admin:rwx,g:
manager:rx ../DATA/
# setfacl -R -m g:assy-dept:rwx,g:
LC-assy:rx assembly/
# setfacl -R -d -m g:assy-dept:
rwx,g:LC-assy:rx assembly/
# setfacl -R -m g:LC-dept:rwx LC/
# setfacl -R -d -m g:LC-dept:rwx LC/
# setfacl -R -m g:mold-dept:rwx,g:
LC-mold:rx molding/
# setfacl -R -d -m g:mold-dept:
rwx,g:LC-mold:rx molding/
# setfacl -R -m g:stamp-dept:rwx,g:
LC-stamp:rx stamping/
# setfacl -R -d -m g:stamp-dept:
rwx,g:LC-stamp:rx stamping/
# setfacl -R -m g:admin:rwx manager/
# setfacl -R -d -m g:admin:rwx
manager/
```

Setelah konfigurasi ACL dilakukan selanjutnya kita lakukan pengecekan ACL pada tiap directory yang sudah dikonfigurasi seperti diatas. Perintah "getfacl"dengan disertai parameter berupa nama directory akan menghasilkan output berupa daftar pengontrolan hak akses pada directory tersebut.

```
# getfacl assembly/
# file: assembly
# owner: root
# group: assy-dept
user::rwx
group::rwx
group:assy-dept:rwx
group:admin:rwx
group:manager:r-x
group:LC-assy:r-x
mask::rwx
other::---
default:user::rwx
default:group::rwx
default:group:assy-dept:rwx
default:group:admin:rwx
default:group:manager:r-x
```



Gambar 1. Pembuatan user1 sampai user8.

```
default:group:LC-assy:r-x
default:mask::rwx
default:other::---

# getfacl LC/
# file: LC
# owner: root
# group: LC-dept
user::rwx
group::rwx
group:LC-dept:rwx
group:admin:rwx
group:manager:r-x
mask::rwx
other::---
default:user::rwx
default:group::rwx
default:group:LC-dept:rwx
default:group:admin:rwx
default:group:manager:r-x
default:mask::rwx
default:other::---

# getfacl manager/
# file: manager
# owner: root
# group: manager
user::rwx
group::rwx
group:admin:rwx
group:manager:r-x
mask::rwx
other::---
default:user::rwx
default:group::rwx
default:group:admin:rwx
```

```
default:group:manager:r-x
default:mask::rwx
default:other::---

# getfacl molding/
# file: molding
# owner: root
# group: mold-dept
user::rwx
group::rwx
group:mold-dept:rwx
group:admin:rwx
group:manager:r-x
group:LC-mold:r-x
mask::rwx
other::---
default:user::rwx
default:group::rwx
default:group:mold-dept:rwx
default:group:admin:rwx
default:group:manager:r-x
default:group:LC-mold:r-x
default:mask::rwx
default:other::---

# getfacl stamping/
# file: stamping
# owner: root
# group: stamp-dept
user::rwx
group::rwx
group:admin:rwx
group:manager:r-x
group:LC-stamp:r-x
group:stamp-dept:rwx
mask::rwx
```

```
other::---
default:user::rwx
default:group::rwx
default:group:admin:rwx
default:group:manager:r-x
default:group:LC-stamp:r-x
default:group:stamp-dept:rwx
default:mask::rwx
default:other::---
```

Setelah menggunakan fitur ACL ini anda harus lebih teliti dalam mengelola hak akses file dalam server anda. Karena dengan perintah ls -l hak akses pada ACL tidak akan muncul dan hanya dapat dilihat dengan perintah getfacl. Untuk membedakan dengan hak akses standar perhatikan karakter terakhir (tanda +) pada hak akses tiap file atau directory. Tanda tersebut mengindikasikan bahwa terdapat hak akses tambahan atau Extended ACL pada directory atau file tersebut.

```
# ls -l /DATA
total 64
drwxrwx---+  9 root       assembly
4096 Jun 25 08:38 assembly
drwxrwx---+ 10 root       logistic
4096 Jun 20 16:04 LC
drwxrwx---+  9 root       molding
4096 Jun 25 11:07 molding
drwxrwx---+ 14 root       manager
4096 Jun 14 13:05 managers
drwxrwx---+ 13 root       stamping
4096 Jun 21 15:50 stamping
```

Untuk melihat perbedaan dengan hak akses standar cobalah untuk menghapus ACL pada salah satu directory yang kita buat.

```
# setfacl -R -b /DATA/molding
# ls -l /DATA
total 64
drwxrwx---+  9 root       assembly
4096 Jun 25 08:38 assembly
drwxrwx---+ 10 root       logistic
4096 Jun 20 16:04 LC
drwxrwx---   9 root       molding
4096 Jun 25 11:07 molding
drwxrwx---+ 14 root       manager
4096 Jun 14 13:05 managers
drwxrwx---+ 13 root       stamping
4096 Jun 21 15:50 stamping
```

Selain parameter–parameter di atas ,masih banyak lagi yang lain dan anda dapat mencobanya sesuai kebutuhan. Untuk lebih detail Anda dapat membaca manual perintah setfacl dan getfacl.■

Sigid [sigidwu@gmail.com]