Bab 4

Resep

Bab ini merupakan inti dari buku ini, menerapkan pola regex dalam tugas sehari-hari. Setiap resep disusun sebagai berikut: **masalah**, yang menyebutkan apa yang ingin diselesaikan; **pola**, yang berisi pola regex dalam format Perl (`/.../` atau `m#...#is`, dan lain sebagainya) dan dapat juga mengandung komentar yang saya tulis dalam gaya shell (`# ...` hingga akhir baris); **contoh kode**, berisi perintah atau program lengkap yang biasanya ditulis dalam Perl dan PHP; serta **diskusi**, yang menjelaskan pola atau program secara lebih mendalam.

Resep-resep yang ada disusun dalam 9 kelompok, di mana prefiks awal nomor resep menyatakan kelompoknya: 1) bilangan; 2) teks/string; 3) file dan direktori; 4) Unix dan sysadmin; 5) HTML, Web, Internet; 6) email; 7) grabbing/scraping; 8) pemrograman; 9) lain-lain.

Pola regex ditulis dalam format dan sintaks Perl, misalnya modifier `s` berarti karakter meta `.` (titik) match newline, bukan s seperti pada Ruby. `\b` berarti karakter meta posisi batas antarkata, bukan backspace seperti pada Tcl, dan seterusnya.

Rata-rata kode program yang diberikan di sini adalah dalam Perl dan PHP. Perl saya pilih karena menulis regex di Perl amat ringkas (dikarenakan adanya operator regex seperti `m//` dan `s///`), sehingga kita bisa lebih banyak berfokus pada regexnya ketimbang baris-baris programnya sendiri. Sementara PHP saya pilih karena banyaknya orang yang menguasai PHP, sehingga bahasa ini saya anggap mewakili mayoritas pembaca.

Catatan: Versi Perl yang saya pakai 5.8.1 di Linux dan 5.8.0 di Windows (menggunakan ActivePerl). Versi PHP yang saya gunakan adalah 4.2.2 dan 4.3.2 di Linux. Setting `register_globals` di `php.ini` adalah On.

1-1 Mencocokkan Pola Bilangan

Masalah

Anda ingin mengecek apakah masukan dari user merupakan/mengandung sebuah bilangan yang valid.

Pola

```
# A. hanya mencocokkan bilangan bulat positif
/\d+/

# B. bilangan bulat positif/negatif, tanda + opsional
/[+-]?\d+/

# C. bilangan bulat atau desimal (dengan tanda titik sebagai
pemisalnyaah desimal)
/[+-]?(?:\d+(?:\.\d*)?|\.\d+)/

# D. bilangan bulat, desimal, atau dalam format eksponen
(misalnya: 1.3e-5)
/([+-]?)(?=\d|\.\d)\d*(\.\d*)?([Ee]([+-]?\d+))?/

# E. pecahan rasional dalam format a/b atau -a/b, a dan b bil
bulat
m#[+-]?\d+(?:/[+-]?\d+)?#
```

Diskusi

Bilangan dapat tampil dalam berbagai format. Selain format-format yang telah disebutkan di atas, bisa saja Anda ingin menerima format bilangan Indonesia (di mana pemisalnyaah desimal adalah koma dan bukan titik). Anda bisa memodifikasi regex C dan D di atas.

Atau bisa juga Anda ingin menerima bilangan yang mengandung pemisalnyaah ribuan (misalnya: dalam format Indonesia 1.200.000). Anda bisa menghapus dulu pemisalnyaah ribuannya misalnyaalnya dengan kode Perl berikut:

```
s/\.//g;
```

Jika Anda ingin mengecek apakah sebuah masukan string hanya berisi bilangan, Anda tinggal menambahkan jangkar `^` dan `$` atau `\A` dan `\z` pada regex untuk memastikan bahwa string tidak mengandung apa-apa yang lain (kecuali newline).

1-2 Ribuan

Masalah

Anda ingin memformat sebuah bilangan agar dipisahkan dengan ribuan, misalnyaalnya 34500000 menjadi 34.500.000.

Contoh kode Perl

```
  1|#!/usr/bin/perl
   |
  3|#
  4|# ribuan.pl
  5|#
   |
  7|sub format_ribuan {
  8|  my $number = shift;
  9|  my $pemisalnyaah_desimal = shift || ","; # default format
```

```
    indonesia
 10|  my $pemisalnyaah_ribuan = shift || ".";  # default format
    indonesia
   |
 12|  $number =~ s/\.(\d+)$//; my $desimal = $1;
 13|  my $bulat = "";
   |
 15|  while ($number =~ s/(?<=\d)(\d\d\d)$//) {
 16|    $bulat = "$bulat$pemisalnyaah_ribuan$1";
 17|  }
   |
 19|  $desimal =~ /\d/ ?
 20|    "$number$bulat$pemisalnyaah_desimal$desimal" :
 21|    "$number$bulat";
 22|}
   |
 24|# test
 25|print format_ribuan(1), "\n",
 26|      format_ribuan(1234), "\n",
 27|      format_ribuan(92302934.01), "\n";
```

Contoh output kode

```
$ ./ribuan.pl
1
1.234
92.934.302,01
```

Diskusi

Perhatikan bagaimana cara fungsi `format_ribuan()` (baris 7-22) melaksanakan tugasnya. Pertama-tama kita membuang dulu koma desimal yang ada dan menyimpannya di variabel `$desimal` (baris 12). Lalu kita memotong bilangan semula (`$number`) tiga angka-tiga angka dari belakang sampai tidak ada ribuan yang tersisa (baris 15) dan menggabungkan kembali tiga-tiga ini di variabel baru `$bulat` (baris 16). Kita tidak bisa menggunakan sekali substitusi regex saja karena titik ribuan disusun dari kanan ke kiri, bukan dari kiri ke kanan.

Perhatikan pula regex di baris 15 kita menggunakan look-behind `(?<=\d)`. Ini agar bilangan seperti "500" tidak diubah menjadi ".500". Tapi "1500" akan diubah menjadi "1.500" karena mengandung 1 di depan 500. Terakhir kita menggabungkan kembali koma desimal (baris 19-21).

Rata-rata implementasi `sprintf()` atau `printf()` biasanya tidak menyediakan cara untuk memformat ribuan, sehingga kita harus membuat sendiri fungsi untuk memformat ribuan ini. Namun jika Anda menggunakan PHP, PHP sudah menyediakan fungsi lain untuk hal ini `number_format()` dan `money_format()`.

1-3 Bilangan Romawi

Masalah

Anda ingin mengetahui apakah sebuah string merupakan/mengandung bilangan romawi yang valid.

Pola

```
/M{0,3}(C[MD]|D?C{0,3})(X[CL]|L?X{0,3})(I[XV]|V?I{0,3})/
```

Diskusi

Bilangan romawi memiliki "digit" `M` (1000), `C` (100), `L` (50), `X` (10), `V` (5), `I` (1). Namun jika Anda menggunakan pola seperti ini saja:

```
/[MCLXVI]+/
```

Anda tidak dapat menjamin sebuah bilangan romawi valid, sebab dalam bilangan romawi huruf-huruf ini harus memiliki susunan tertentu (misalnya: `MXXM` biasanya dianggap tidak valid; 1980 harus ditulis sebagai `MCMLXXX`).

Pola di atas hanya dapat dipakai untuk bilangan dari 1 hingga 3999; namun untuk sebagian besar aplikasi sudah cukup sebab biasanya angka romawi dipakai untuk menomori poin/bab (1, 2, 3, …) atau untuk tahun (misalnya: 2000, 2004, dan lain sebagainya).
Sebagai catatan tambahan, untuk notasi 5000 biasanya digunakan `V` dengan garis horizontal tunggal di atasnya, untuk 10.000 `X` dengan garis tunggal di atas, dan seterusnya.

Jika Anda ingin mengecek sebuah string *hanya* terdiri dari bilangan romawi, jangan lupa beri jangkar `^` dan `$` atau `\A` dan `\z` pada pola. Jika Anda ingin bilangan romawi dalam huruf kecil (misalnya: `mmiv`) juga dianggap valid, Anda bisa menambahkan modifier `i` atau mengkonversi dulu stringnya menjadi huruf besar.

Bagaimana dengan konversi dari desimal ke/dari romawi? Saya serahkan sebagai PR untuk Anda, karena toh problem tersebut tidak terlalu relevan dengan regex. Sudah ada banyak librari atau contoh kode untuk melakukannya; bisa Anda cari di Internet.

1-4 Nomor Cantik

Masalah

Anda ingin mengetahui apakah sebuah angka memiliki pola keteraturan tertentu, misalnyaalnya mengandung empat atau lebih digit yang sama secara berturutan (misalnya: 8888), atau mengandung deretan empat atau lebih angka yang bersusun (misalnya: 3456), dan lain sebagainya.

Contoh kode PHP

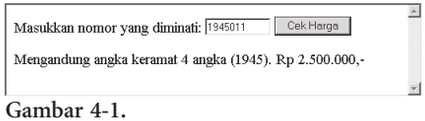
```
 1|<?
  |
 3|//
 4|// nomor-cantik.php
 5|//
  |
 7|if (!isset($nomor)) $nomor = "";
  |
 9|echo "
10|  <form method=GET>
11|  Masukkan nomor yang diminati:
12|    <input name=nomor size=10 maxlength=7 value=\"",
13|                  htmlentities($nomor),"\">
14|  <input type=submit value=\"Cek Harga\">
15|  </form>
16|";
  |
18|if (strlen($nomor)) {
19|  if (!preg_match("/^\d+$/", $nomor))
20|    echo "Bukan angka!";
  |
22|  elseif (!preg_match("/^\d{7}$/", $nomor))
23|    echo "Masukkan 7 angka!";
  |
25|  elseif (preg_match('/(.)\1\1\1/', $nomor, $m))
26|    echo "Mengandung deret empat angka yang sama ($m[0]). Rp
   10.000.000,-";
  |
28|  elseif (preg_match('/(.)(.)\1\2\1\2/', $nomor, $m))
29|    echo "Mengandung tiga pasang angka berurutan ($m[0]). Rp
   10.000.000,-";
```



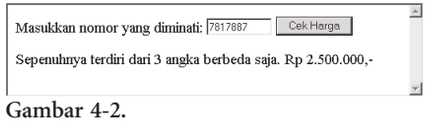
```
  |
31|  elseif (preg_match('/(.)(.).+\1\2\1\2/', $nomor, $m) ||
32|           preg_match('/(.)(.)\1\2.+\1\2/', $nomor, $m))
33|    echo "Mengandung tiga pasang angka berurutan
   ($m[1]$m[2]).
34|          Rp 7.500.000,-";
  |
36|  elseif (preg_match('/17845|   # 17-agu-1945
37|                       170845|  # idem
38|                       1781945  # idem
39|                      /x', $nomor, $m))
40|    echo "Mengandung angka keramat 5/lebih angka ($m[0]).
41|          Rp 7.500.000,-";
  |
43|  elseif (count(array_unique(preg_split("//", $nomor, -1,
44|
   PREG_SPLIT_NO_EMPTY))) <= 2)
45|    echo "Sepenuhnya terdiri dari 2 angka berbeda saja. Rp
   7.500.000,-";
  |
47|  elseif (preg_match('/(.)(.)\1\2/', $nomor, $m))
48|    echo "Mengandung dua pasang angka berurutan ($m[0]). Rp
   7.500.000,-";
  |
50|  elseif (preg_match('/(.)(.)(.)(.)/', $nomor, $m) &&
51|           (($m[2] == $m[1]+1 && $m[3] == $m[2]+1 && $m[4] ==
   $m[3]+1) ||
52|            ($m[2] == $m[1]-1 && $m[3] == $m[2]-1 && $m[4] ==
   $m[3]-1)))
53|    echo "Mengandung deret empat angka bersusun ($m[0]). Rp
   7.500.000,-";
  |
55|  elseif (preg_match('/(.)(.)(.)/', $nomor, $m) &&
56|           (($m[2] == $m[1]+1 && $m[3] == $m[2]+1) ||
57|            ($m[2] == $m[1]-1 && $m[3] == $m[2]-1)))
58|    echo "Mengandung deret tiga angka bersusun ($m[0]). Rp
   5.000.000,-";
  |
60|  elseif (preg_match('/(.)\1\1/', $nomor, $m))
61|    echo "Mengandung tiga angka yang sama ($m[0]). Rp
   5.000.000,-";
```

```
   |
63|  elseif (count(array_unique(preg_split("//", $nomor, -1,
64|
   PREG_SPLIT_NO_EMPTY))) <= 3)
65|    echo "Sepenuhnya terdiri dari 3 angka berbeda saja. Rp
   2.500.000,-";
   |
67|  elseif (preg_match('/1745|   # 17-agu-1945
68|                      1945    # idem
69|                     /x', $nomor, $m))
70|    echo "Mengandung angka keramat 4 angka ($m[0]).
71|          Rp 2.500.000,-";
   |
73|  elseif (preg_match('/(.)(.).+\1\2/', $nomor, $m))
74|    echo "Mengandung dua pasang angka tidak berurutan
   ($m[1]$m[2]).
75|          Rp 1.000.000,-";
   |
77|  else
78|    echo "Bukan nomor cantik. Rp 250.000,-";
79|}
   |
81|?>
```

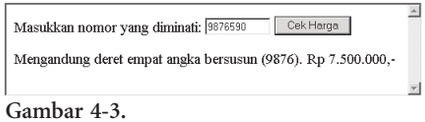
Contoh output



Gambar 4-1.



Gambar 4-2.



Gambar 4-3.

Gambar 4-4.

Diskusi

Nomor cantik biasanya dijual oleh operator ponsel atau perusahaan telepon dengan harga "cantik" (baca: mahal) juga. Telkomsel misalnyaalnya, mematok harga dari 500 ribu s.d. 1 juta untuk nomor yang "agak cantik" hingga 10 juta rupiah untuk nomor yang "benar-benar cantik". Nomor cantik selain diminati oleh perusahaan

yang ingin nomor kontaknya mudah diingat, juga oleh perorangan yang mempercayai kombinasi nomor tertentu membawa keberuntungan.

Dalam mencari pola-pola nomor cantik ini kita menggunakan berbagai macam regex dan kadang dibantu juga dengan rutin lain. Kita perlu melakukan pengecekan dari pola yang terpanjang/harga termahal hingga pola yang pendek/harga yang termurah. Jika tidak, angka seperti 4208888 dapat ditentukan hanya mengandung tiga digit yang sama dengan harga Rp 5.000.000,- padahal seharusnya mengandung empat digit yang sama dengan harga Rp 10.000.000,-.

Mari perhatikan beberapa pengecekan yang menarik. Di baris 25 misalnyaalnya, kita ingin mengecek empat angka yang sama berturutan (misalnya: 3333, 4444). Kita perlu menggunakan kutip tunggal untuk mengapit pola regex karena PCRE mengharapkan dua karakter \ dan 1 dan bukannya "\1" yang diubah oleh PHP menjadi ASCII 1.

Mulai baris 25 kita mencocokkan angka dengan `.` saja dan bukan `\d` karena toh di baris 19 kita sudah memfilter `$nomor` harus berisi bilangan saja.

Di baris 36-39 dan 67-69 kita menggunakan modifier `x` untuk memberi komentar arti dari setiap angka keramat.

Di baris 43 kita menggunakan `preg_split()` untuk mengubah bilangan menjadi array berisi tiap digit. Lalu kita membuang angka yang dobel-dobel, sehingga jika panjang array hanyalah 2 elemen, maka kita tahu bahwa hanya ada 2 angka berbeda pada nomor cantik (misalnya: 1221221).

Di baris 50-52 dan 55-57 kita menggunakan bantuan operator aritmetika jika pengecekan ini dilakukan dengan regex saja akan cukup panjang pola yang harus dibuat:

```
/0123|1234|2345|3456|4567|5678|6789|7890/
```

tapi kelebihan pola di atas adalah dapat mendeteksi 7890 sebagai empat deret sementara dengan cara aritmetika tidak bisa karena 0 bukanlah 9+1. Tentu saja, cukup mudah bagi Anda untuk memodifikasi programnya agar bisa juga mendeteksi hal ini.

Tentu saja Anda bisa memodifikasi program PHP di atas dengan menambahkan rutin-rutin lain untuk memeriksa "kecantikan-kecantikan" yang lain, misalnyaalnya mengecek apakah terdapat angka 8 dalam jumlah tertentu (karena angka 8

dianggap hoki oleh sebagian orang), terdapat deretan angka keramat tertentu (misalnya: 13298 melambangkan tanggal 13 Feb 1998 yaitu kelahiran putra si peminat nomor, dan lain sebagainya).

Satu PR bagi Anda yang berminat, yaitu mengecek apakah sebuah nomor telepon bisa membentuk kata. Misalnya, 7435763 dapat membentuk 7-HELPME karena setiap digit pada nomor telepon dapat diwakili satu atau lebih huruf (4 dapat diwakili G, H, I; 9 diwakili W, X, Y, Z; dan seterusnya). Solusi ini tidak bertumpu pada regex, tapi regex dapat membantu mengecek apakah sebuah nomor tidak dapat dijadikan kata, yaitu nomor telepon yang [terlalu banyak] mengandung angka 0 dan 1, karena 0 dan 1 tidak diwakili huruf.
Bagi Anda yang penasaran, dapat melihat programnya di CD, `phone2word.pl`.

2-1 Kata

Masalah

Anda ingin menemukan/mengekstrak/menghitung kata-kata dalam sebuah teks.

Pola

```
# sederhana
/\b(\w+)\b/

# mencakup juga kata ulang
/\b(\w+)-\1\b/

# mencakup juga kata berapostrof
/\b(\w+'\w*|\w*'\w+|\w+)\b/

# mencakup juga kata yang mengandung strip (kata ulang
berimbuhan,
# dua kata gabungan, dan lain sebagainya)
/\b([\w-]+)\b/

# sepenuhnya mengandalkan \b regex
/\b(.+?)\b/

# cara 'kasar', menganggap semua kumpulan non-whitespace sebagai
kata
/(\S+)/
```


Contoh kode Perl (implementasi sederhana wc)

```
 1|#!/usr/bin/perl
  |
 3|#
 4|# wc.pl
 5|#
  |
 7|use Tie::InsertOrderHash;
  |
 9|$current_file = "";
10|tie %files, 'Tie::InsertOrderHash';
  |
12|while ($line = <>) {
13|  # telah berganti file
14|  if ($ARGV ne $current_file) {
15|    $current_file = $ARGV;
16|    $files{$current_file} = { lines => 0, words => 0, bytes
   => 0 };
17|  }
  |
19|  # jangan hitung baris kosong di akhir file
20|  unless (eof(ARGV) && $line !~ /\S/) {
21|    $files{$current_file}{lines}++;
22|    $files{$current_file}{words}++ while $line =~ /\S+/g;
23|  }
  |
25|  $files{$current_file}{bytes} += length($line);
26|}
  |
28|# total
29|$total_lines = $total_words = $total_bytes = 0;
  |
31|for $file (keys %files) {
32|  printf "%7d %7d %7d %s\n",
33|    $files{$file}{lines},
34|    $files{$file}{words},
35|    $files{$file}{bytes},
36|    ($file eq '-' ? "" : $file);
37|  $total_lines += $files{$file}{lines};
38|  $total_words += $files{$file}{words};
```

```
 39|  $total_bytes += $files{$file}{bytes};
 40|}
   |
 42|# tampilkan  total
 43|if (keys (%files) > 1) {
 44|  printf "%7d %7d %7d %s\n",
 45|    $total_lines,
 46|    $total_words,
 47|    $total_bytes,
 48|    "total";
 49|}
```

Contoh output program Perl

```
[steven@builder regex]$ wc *
      7      29     219 2.pl
     10      37     229 3.pl
     81     280    2708 nomor-cantik.php
     81     351    3113 nomor-cantik.php.txt
     60     213    1197 phone2word.pl
     60     247    1497 phone2word.pl.txt
      6      28     198 re1.pl
     27      69     565 ribuan.pl
     27      88     700 ribuan.pl.txt
     10      67     457 teks1.txt
     49     128    1015 wc.pl
     48     128    1016 wc.pl~
     30      94     728 wc.pl.txt
    496    1759   13642 total
[steven@builder regex]$ ./wc.pl *
      7      29     219 2.pl
     10      37     229 3.pl
     81     280    2708 nomor-cantik.php
     81     351    3113 nomor-cantik.php.txt
     60     213    1197 phone2word.pl
     60     247    1497 phone2word.pl.txt
      6      28     198 re1.pl
     27      69     565 ribuan.pl
     27      88     700 ribuan.pl.txt
     10      67     457 teks1.txt
     49     128    1015 wc.pl
```



```
     48      128     1016 wc.pl~
     30       94      728 wc.pl.txt
    496     1759    13642 total
```

Contoh kode PHP (statistik teks)

```
  1|<?
   |
  3|//
  4|// statistik-teks.php
  5|//
   |
  7|if (!isset($submit)) $submit = "";
  8|if (!isset($teks)) $teks = "";
   |
 10|echo "
 11|  <form method=POST>
 12|  <textarea name=teks cols=80
   rows=11>",htmlentities($teks),"</textarea>
 13|  <br>
 14|  <input type=submit name=submit>
 15|  </form>
 16|";
   |
 18|if ($submit) {
 19|  preg_match_all("/^(.*\n)/m", $teks, $baris,
   PREG_SET_ORDER);
 20|  preg_match_all("/\b(\w+'\w*|\w*'\w+|\w+)\b/", $teks, $kata,
 21|                 PREG_SET_ORDER);
   |
 23|  $jumlah_baris = 0;
 24|  $jumlah_baris_takkosong = 0;
 25|  foreach ($baris as $b) {
 26|    $jumlah_baris++;
 27|    if (preg_match("/\S/", $b[0])) $jumlah_baris_takkosong++;
 28|  }
   |
 30|  $jumlah_kata = 0;
 31|  $kata_unik = array();
 32|  foreach($kata as $k) {
 33|    $jumlah_kata++;
```

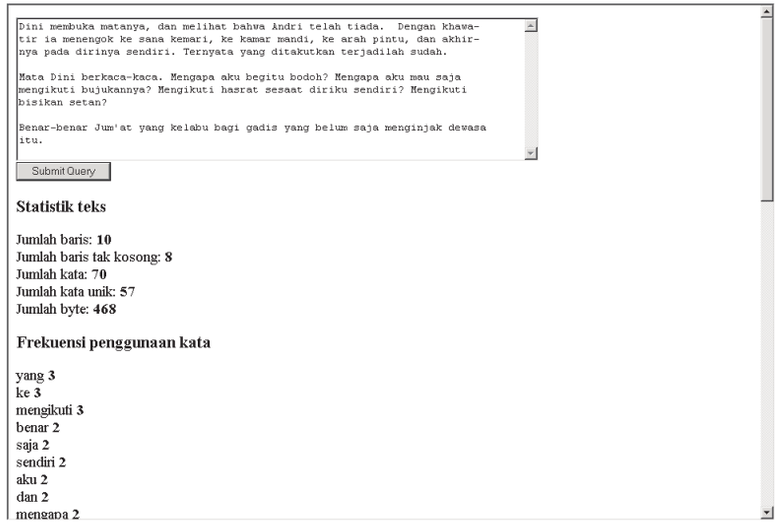
```
34|    $key = strtolower($k[0]);
35|    if (!isset($kata_unik[$key])) $kata_unik[$key] = 0;
36|      $kata_unik[$key]++;
37|  }
38|  $jumlah_kata_unik = count($kata_unik);
   |
40|  $jumlah_byte = strlen($teks);
   |
42|  echo "
43|    <h3>Statistik teks</h3>
44|    <p>Jumlah baris: <b>$jumlah_baris</b><br>
45|       Jumlah baris tak kosong: <b>$jumlah_baris_takkosong</
   b><br>
46|       Jumlah kata: <b>$jumlah_kata</b><br>
47|       Jumlah kata unik: <b>$jumlah_kata_unik</b><br>
48|       Jumlah byte: <b>$jumlah_byte</b>
   |
50|    <h3>Frekuensi penggunaan kata</h3>
51|  ";
   |
53|  arsort($kata_unik);
54|  foreach($kata_unik as $k => $f) {
55|    echo "$k <b>$f</b><br>";
56|  }
57|}
   |
59|?>
```

Contoh output program PHP

Gambar 4-5.

Diskusi

Yang dimaksud "kata" kadang-kadang bermacam-macam, bergantung definisi dan interpretasi si pembuat program. Jika Anda menghitung jumlah kata pada sebuah teks dengan perintah wc di Unix, Microsoft Word, dan dengan editor lain, hasilnya mungkin berbeda-beda, bergantung pada bagaimana setiap program menganggap apa yang dimaksud dengan "kata". Pola regex pada resep ini memberikan beberapa alternatif yang mungkin bisa Anda coba. Tidak ada satu yang benar atau salah, semuanya bergantung pilihan Anda. Jika Anda menggunakan `\w+` saja, maka Anda akan menghitung kata ulang seperti "kura-kura" bahkan kata-kata berapostrof seperti "Jum'at" sebagai 2 kata. Jika ini tidak Anda inginkan, maka Anda bisa



Gambar 4-5.

menggunakan `\S+`. Tapi konsekuensinya, jika ada tanda baca setelah huruf terakhir kata seperti "adalah;" maka tanda baca tersebut akan disertakan sebagai bagian dari kata. Bahkan jika ada tanda baca yang berdiri sendiri seperti "apa ?", maka tanda baca tersebut akan dianggap sebuah kata juga. Atau apakah teks perlu dipraproses dulu misalnyaalnya.

Selain isu di atas, kadang-kadang Anda juga harus memutuskan isu seperti pelipatan kata (text wrapping/hyphenation). Misalnyaalnya kalau ada teks seperti di bawah ini:

```
Dini membuka matanya, dan melihat bahwa Andri telah tiada.
Dengan khawatir ia menengok ke sana kemari, ke kamar mandi,
ke arah pintu, dan akhirnya pada dirinya sendiri. Ternyata
yang ditakutkan terjadilah sudah.
```

apakah kata-kata `khawatir` dan `akhirnya` ingin disatukan dulu atau dibiarkan dihitung sebagai dua kata? Semua tergantung pada Anda tentunya untuk memilih kelakuan yang diinginkan.

wc.pl. Mari membahas program pertama pada resep ini yaitu wc.pl. Perintah ini mencoba mengimplementasikan perintah Unix **wc** ("word count"). Perintah ini dapat menerima teks dari stdin atau dari beberapa file yang disebutkan di argumen.

Operator <> Perl memberi kemudahan pada kita karena kita dapat mengambil semuanya dalam satu looping, tak peduli masukan dari stdin atau dari satu/lebih file. Di baris 14 kita mengecek apakah telah terjadi pergantian file; jika ya maka kita perlu mereset counter baris, kata, dan byte karena wc menghitung per file.

Untuk menghitung jumlah kata (baris 22), kita menggunakan regex `\S+`. Regex ini nampaknya menghasilkan kelakuan yang sama dengan wc Unix (artinya, menghasilkan hasil jumlah kata yang sama). Jika digunakan `\w+` maka jumlah kata yang terhitung jadi lebih banyak.

Di baris 20 kita melakukan pengecekan apakah ini baris terakhir file yang kosong atau hanya terdiri dari spasi/tab (dengan kata lain, tidak ada non-whitespace, dan regex `/\S/` akan gagal match). Jika memang kosong, kita tidak menghitungnya sebagai baris (tapi tetap menghitung jumlah bytenya, baris 25).

Kelakuan ini sama dengan kelakuan wc Unix. Output program adalah seperti diperlihatkan di atas. Field pertama adalah jumlah baris, field kedua jumlah kata, field ketiga jumlah byte. Jika ada lebih dari satu file, kita menampilkan totalnya (baris 43-49).

Tentu saja program `wc.pl` yang singkat ini tidak sepenuhnya mengimplementasi perintah wc. Ada kemampuan lain `wc` yaitu menerima opsi seperti `-l` (hanya mengoutput jumlah baris), `-w` (hanya mengoutput jumlah kata), lalu dapat menghitung jumlah *karakter* (yang tidak selalu sama dengan jumlah *byte* untuk set karakter seperti Unicode UTF8 atau UTF16). Mungkin bisa jadi PR bagi Anda yang tertarik.

statistik-teks.php. Mari membahas program kedua, kali ini dalam PHP. Kali ini kita menggunakan regex:

```
/\b(\w+'\w*|\w*'\w+|\w+)\b/
```

dan bukannya `/\S+/` karena kita tidak ingin string seperti `"?"`, `"..."`, atau `"itu."` (tanda baca titik terikutsertakan) dianggap sebagai kata. Regex di atas berarti deretan apostrof dapat terdapat di tengah atau di akhir kata (`\w+'w*`) atau di tengah/awal kata (`\w*'\w+`). Tentu saja kita juga memberikan alternatif `\w+` saja.

Jika tidak, hanya kata-kata berapostrof saja yang akan dihitung! Juga perlu dicatat bahwa regex ini belum mencocokkan kata yang mengandung 2 atau lebih apostrof. Jika Anda ingin kata seperti itu juga tertangkap, silakan modifikasi pola regexnya. Baris 19 mengambil semua kata dan memasukkan ke dalam array `$baris`.

Baris 23-28 lalu menghitung jumlah baris yang ada termasuk jumlah baris yang yang tidak kosong. Baris 20-21 mengambil semua kata yang ada pada teks dan memasukkan dalam array `$kata`.

Baris 30-38 lalu menghitung jumlah kata yang ada, termasuk menghitung jumlah kata unik (setiap kata yang sama hanya dihitung sekali). Untuk melakukan yang terakhir ini, kita menyimpan setiap kata di dalam key array setelah terlebih dahulu mengkonversikannya menjadi huruf kecil. Ini agar "Itu" dan "itu" tidak dianggap dua kata unik.

Setelah perhitungan selesai, selanjutnya tinggal menampilkan hasil (baris 42-56). Di baris 53-56 kita menampilkan frekuensi penggunaan kata, mulai dari yang paling sering digunakan hingga yang terjarang.

2-2 Kata Ulang

Masalah

Anda ingin mengetahui/menangkap kata ulang dalam teks.

Pola

```
# A. hanya kata ulang murni (misalnya: kura-kura, Kupu-kupu)
/\b(\w+)-\1\b/i

# B. kata ulang berimbuhan (misalnya: selama-lamanya, bersenang-
senang)
/\b\w*(\w{3,})-\1\w*\b/i
```

Diskusi

Mengapa di pola B kita membatasi minimal kata dasarnya 3 huruf ke atas? Karena jika tidak, kata seperti "berasa-asin" akan cocok juga, di mana `\1` adalah "a". Tentu hal ini tidak diinginkan.

Kalau tidak salah ingat, dalam mata pelajaran bahasa Indonesia waktu sekolah menengah dahulu, ada beberapa macam lagi kata ulang. Yakni:
1) kata ulang salin suara, di mana kata ulangnya mengalami perubahan (misalnya: ramah-tamah atau warna-warni);
2) kata ulang salin suara berimbuhan (misalnya: beramah-tamah);
3) kata ulang tiga kata salin suara, terutama di bahasa Sunda (misalnya: dar-der-dor, tat-tit-tut). Agak sulit mencari pola kata-kata seperti ini dengan menggunakan regex.

2-3 Palindrom

Masalah
Anda ingin mengetahui apakah sebuah kata merupakan palindrom atau bukan.

Pola

```
# dapat mencocokkan hingga palindrom 11 huruf
/\b(\w)(\w)(\w)(\w)(\w+)\w?\5\4\3\2\1\b     | # 10-11 huruf
  \b(\w)(\w)(\w)(\w)\w?\4\3\2\1\b           | # 8-9 huruf
  \b(\w)(\w)(\w)\w?\3\2\1\b                 | # 6-7 huruf
  \b(\w)(\w)\w?\2\1\b                       | # 4-5 huruf
  \b(\w)\w?\1\b                             | # 2-3 huruf
  \b\w\b                                    | # 1 huruf
/xi
```

Diskusi
Seperti pernah dijelaskan dalam latihan di Bab 2, palindrom adalah kata yang jika dibalik huruf per huruf hasilnya kata yang sama, contohnya: “bab”, “malam”, “kodok”. Dengan definisi ini, semua kata satu huruf adalah palindrom. Semakin panjang palindrom, semakin sulit dan jarang. Tanpa mencontek kamus, saya saat ini bahkan tidak bisa mencari palindrom yang 7 huruf ke atas (kecuali mungkin “kitikitikitik” atau “hahehihehah”). Ironisnya, kata palindrom bukan palindrom.

2-4 Baris

Masalah
Anda ingin memecah/mengambil/menghitung baris dalam teks.

Pola

```
# A. dengan modifier m jadi mudah
/^.*\n/m

# B. sama dengan A, tapi jika baris terakhir tidak diakhiri
newline, regex
# berikut masih bisa mengambilnya
/^.*(\n|\z)/m

# C. tanpa modifier m
/([^\n]+\n?)/
```


Diskusi
Contoh penggunaan regex ini ada di resep 2-1 di skrip `statistik-teks.php`. Modifier m berguna agar karakter meta ^ cocok dengan awal setiap baris, bukan hanya awal string saja. Idealnya memang setiap baris, termasuk baris terakhir, diakhiri dengan newline. Dengan kata lain, newline haruslah menjadi karakter terakhir setiap file teks. Namun, kadang-kadang hal ini dilanggar. Dengan regex B, kita membuat alternatif (`\n|\z`) yang berarti “jika newline tidak ditemukan, cocokkan dengan akhir string”. Tanpa pola ini (seperti di regex A), baris terakhir yang tidak diakhiri newline tidak akan tertangkap.

Catatan: jika Anda mengambil teks sudah baris demi baris, seperti pada contoh kode `wc.pl` di resep 2-1, tentu saja Anda cukup membuat counter di setiap kali pemba caan. Di akhir loop, counter akan berisi jumlah baris yang Anda baca.

2-5 Memecah Teks Per Huruf

Masalah
Anda ingin memecah sebuah teks menjadi huruf per huruf.

Contoh kode Perl

```
@huruf = split //, "teks...";
```

Contoh kode PHP

```
$huruf = preg_split("//", "teks...", -1, PREG_SPLIT_NO_EMPTY);
```

Contoh kode Python

```
# import re dulu tentunya
huruf = [i for i in re.split("(.)", "teks...") if i!='']
```

Contoh kode Ruby

```
huruf = "teks...".split //
```

Diskusi
Kelakuan split regex di berbagai bahasa agak berbeda. Contohnya, di PHP, jika kita tidak menyebutkan flag `PREG_SPLIT_NO_EMPTY`, maka akan dihasilkan dua elemen kosong di ujung depan dan belakang. Di Python juga split dengan regex kosong tidak memberikan hasil sebagaimana di bahasa-bahasa lainnya.

Perlu dicatat bahwa cara regex bukanlah yang paling efisien/cepat untuk mengekstrak huruf-huruf dari string, meskipun biasanya paling nyaman dan ringkas. Pada bahasa-bahasa di mana setiap karakter pada string dapat dialamati dengan subskrip, misalnyaalnya di C atau Python atau Ruby, Anda dapat menggunakan operator `[]` (misalnya: `kata[3]` untuk mengambil karakter keempat). Di Ruby juga misalnyaalnya Anda dapat mengiterasi setiap byte dengan metode `each_byte`. Atau Anda juga bisa menggunakan fungsi `substr()`.

2-6 Membuang whitespace

Masalah

Anda ingin membuang spasi/tab/newline ekstra/yang tidak dibutuhkan.

Contoh kode Perl

```
# A. membuang whitespace di awal teks
s/^\s+//;

# B. membuang whitespace di akhir teks
s/\s+$//;

# C. A dan B sekaligus, belum tentu lebih cepat dari 2 operasi A
+ B secara seri
s/^\s+(.+?)\s+$/$1/;

# D. membuang baris kosong
s/^\s*$//mg;

# E. sama dengan D, menggunakan lookahead dan tanpa modifier m
s/(\n\s*(?=\n))+//g;

# F. mengganti 2 atau lebih spasi menjadi hanya 1
s/ {2,}/ /g;

# G. mengganti tab, newline, atau spasi ekstra menjadi 1 spasi
s/\s{2,}/ /sg;
```

Contoh kode PHP

```
// implementasi ltrim()
function my_ltrim($str) {
  return preg_replace("/^\s+/", "", $str);
}

// implementasi rtrim()
function my_rtrim($str) {
  return preg_replace("/\s+$/", "", $str);
}
```

Diskusi

Whitespace ekstra kadang-kadang tidak diinginkan kehadirannya. Dengan regex hal ini mudah sekali dilakukan, bisa dilihat dari kode-kode Perl dan contoh kode PHP untuk mengimplementasikan `ltrim()` dan `rtrim()`–hanya perlu 1 baris!

Pola G biasanya dipakai untuk menyatukan teks beberapa baris menjadi satu baris panjang.

2-7 Huruf Besar dan Kecil

Masalah

Anda ingin mengkonversikan dari huruf besar dan huruf kecil.

Contoh kode Perl

```
# A. ubah ke huruf kecil
s/([a-z]+)/uc $1/e;

# B. ubah ke huruf kecil, sama dengan lc()
s/([A-Z]+)/lc $1/e;

# C. ubah huruf pertama ke huruf kecil, sama dengan ucfirst()
s/^(.)/uc $1/;

# D. ubah huruf pertama ke huruf kecil, sama dengan lcfirst()
s/^(.)/lc $1/;

# E. ubah huruf kecil jadi besar, besar jadi kecil
s/([A-Za-z])/$1 ge "a" && $1 le "z" ? uc($1) : lc($1)/eg;

# F. Title Case
s/\b(\w)(.+?)\b/uc($1).lc($2)/eg;
```

Contoh kode PHP:

```
<?

//
// title-case.php
//

function title_case($str) {
  return preg_replace('/\b(\w)(.+?)\b/e',
                      "strtoupper('\\1').strtolower('\\2')",
$str);
}

// test
echo title_case("ATOMIC TRANSACTIONS by nancy lynch");

?>
```

Output contoh kode PHP

```
Atomic Transactions By Nancy Lynch
```

Diskusi

Konversi huruf besar dan huruf kecil merupakan sesuatu yang amat fundamental. Untuk mayoritas kasus kita sebetulnya tidak perlu menggunakan regex, karena praktis semua bahasa pemrograman menyediakan fungsi seperti `uc()`/`strtoupper()`/`upcase`/`lc()`/`strtrtolower()`/`downcase`/dan lain sebagainya. Tapi kadang-kadang untuk melakukan `ucfirst()` atau title case (Di Mana Hanya Huruf Pertama Dari Tiap Kata Yang Dikapitalkan, Yang Lain Huruf Kecil) tidak ada fungsinya; dalam kasus-kasus ini kita bisa menggunakan regex. Perl misalnya tidak menyediakan title case. PHP menyediakan lewat fungsi `mb_convert_case()`. Sebagai tantangan untuk Anda, bagaimana caranya jika kita ingin Title Case tapi memperhatikan kata-kata depan (di, ke, dari)? Kata-kata ini diubah menjadi berawalan huruf besar jika di awal teks:

```
Dari Sabang Sampai Merauke
```

tapi tetap jadi huruf kecil jika ada di tengah atau akhir:

```
Antara Anyer dan Jakarta, Kita Jatuh Cinta
```


Catatan: Anda dapat membuatnya di Perl dengan `s/PAT/CODE/eg` dengan disertai counter `$i` yang dimulai dari 1 dan ditambah di setiap kali `CODE` dieksekusi. Berarti, `$i=1` menunjukkan kata pertama yang harus dikapitalkan tak peduli kata depan atau bukan; dan `$i>1` yang menunjukkan kita harus mengecek dulu keberadaan kata depan sebelum mengkapitalkan kata.

Atau, Anda juga dapat memecah dulu teks menjadi array kata.

2-8 Melipat Teks

Masalah

Anda memiliki teks yang panjang, Anda ingin membelahnya menjadi beberapa baris di mana tiap baris maksimum 72 karakter panjangnya.

Contoh kode Perl

```
 1|#!/usr/bin/perl
  |
 3|#
 4|# textwrap.pl
 5|#
  |
 7|use Getopt::Std;
  |
 9|$opts{l} = 72;   # default panjang baris 72 baris
  |
11|getopt('l', \%opts);
12|$l2 = $opts{l} - 1;
  |
14|$str = "";
  |
16|while (<>) {
17|  s/\s+$//s;
18|  $str .= $_;
  |
20|  while (length($str) > $opts{l}) {
21|    if ($str =~ s/^(.{1,$l2})(\s+)(\S+)/$3/o) {
22|      print $1;
23|      if (length("$1$2$3") >= $opts{l}) {
24|        print "\n";
25|      } else {
```

```
26|        print $2;
27|      }
28|    } elsif ($str =~ s/^(.{$l2})//o) {
29|      print "$1-\n";
30|    }
31|  }
32|}
33|print $str, "\n";
```

Diskusi

Program `textwrap.pl` di atas menerima masukan sebaris-sebaris lalu menggabungkan tiap baris yang diterimanya ke `$str` (setelah terlebih dahulu melucuti newline pada baris tersebut).

Di baris 21, kita mengecek apakah sebuah kata bisa dipecah pada posisi dekat 72 kolom. Variabel `$l2` bernilai 72-1 = 71 sehingga subpola `.{1,71}` berarti cocokkan sebanyak mungkin hingga 71 karakter. Lalu ambil juga spasi (`\s+`) dan sebuah kata berikutnya.

Jika ketiga subpola ini panjangnya lebih dari maksimum baris (72 karakter), maka kita mencetak newline setelah subpola pertama (baris 24). Jika tidak, kita hanya mencetak spasi (baris 26).

Jika ternyata pengecekan di baris 21 gagal, ini berarti kemungkinan ada sebuah kata yang amat panjang (misalnya, deretan 80 karakter "= "). Kita lalu memaksakan memotong pada posisi 71 karakter dan menyisipkan tanda "-" diikuti newline. Hasilnya adalah, tidak ada baris yang dicetak melebihi 72 karakter, sesuai harapan kita.

Catatan: kehadiran karakter tab ("\t", "\011", "\x09") akan mengacaukan perhitungan program sederhana kita karena karakter ini memiliki panjang yang bervariasi, bisa 1 hingga 8 karakter. Untuk menghadapi tab, ada beberapa alternatif solusi.

Pertama, kita dapat mengubah dulu semua tab pada string masukan menjadi spasi (lihat resep 2-9). Kedua, kita dapat mengganti fungsi `length()` di baris 20 dan 23 menjadi fungsi custom yang dapat menghitung panjang baris dengan memperhitungkan tab.

Catatan: Perl menyediakan modul untuk melipat teks yaitu `Text::Wrap`. PHP juga menyediakan fungsi `wordwrap()`.

2-9 Konversi Tab

Masalah

Anda ingin mengkonversi tab menjadi spasi atau sebaliknya

Contoh kode Perl (konversi tab menjadi spasi)

```
 1|#!/usr/bin/perl
  |
 3|#
 4|# tab2space.pl
 5|#
  |
 7|use Getopt::Std;
 8|use POSIX 'ceil';
  |
10|$opts{w} = 8;   # lebar tab default
11|getopt('w', \%opts);
  |
13|while (<>) {
14|  while (/\t/) {
15|    s[^([^\t]*)\t]
16|     [ $1 . " " x (ceil((length($1)+1)/8)*$opts{w} -
   length($1)) ]e;
17|  }
18|  print;
19|}
```

Contoh kode Perl (konversi spasi menjadi tab)

```
 1|#!/usr/bin/perl
  |
 3|#
 4|# space2tab.pl
 5|#
  |
 7|use Getopt::Std;
 8|use POSIX 'ceil';
  |
10|$opts{w} = 8;   # lebar tab default
11|getopt('w', \%opts);
  |
```

```
13|while (<>) {
14|  chomp;
  |
16|  $pos = 0;
17|  while (/(\t| +|[^ ]+)/g) {
18|    my $token = $1;
  |
20|    if ($token eq "\t") {
21|      $pos = ceil(($pos+1)/$opts{w}) * $opts{w};
22|      print $token;
23|    } elsif ($token !~ / {2,}/) {
24|      $pos += length($token);
25|      print $token;
26|    } else {
27|      # berapa posisi jika ditambah spasi
28|      $pos2 = $pos + length($token);
  |
30|      # berapa lebar jika kita pakai tab lalu sisanya pakai
   spasi
31|      $ntab = 0; $nspace = 0;
32|      $pos3 = $pos;
33|      while ($pos3 < $pos2) {
34|        $pos3b = ceil(($pos3+1)/$opts{w})*$opts{w};
35|        if ($pos3b <= $pos2) {
36|          $ntab++;
37|        } else {
38|          $nspace += $pos2-$pos3;
39|        }
40|        $pos3 = $pos3b;
41|      }
42|      $tabnspace = ("\t" x $ntab) . (" " x $nspace);
  |
44|      # apakah kita bisa mengirit jumlah karakter memakai
   tab?
45|      if (length($tabnspace) < length($token)) {
46|        print "$tabnspace";
47|      } else {
48|        print "$token";
49|      }
50|      $pos = $pos2;
51|    }
```



```
52|  }
  |
54|  print "\n";
55|}
```

Diskusi

Tab adalah karakter yang seringkali mengundang perdebatan religius terutama seputar gaya indentasi. Pihak yang tidak menyukai tab berargumen bahwa karakter tab dapat diset berbeda lebarnya sehingga membuat penampilan program dapat berbeda-beda (tidak sesuai tampilan original), apalagi jika dicampur dengan spasi (bisa mencang-mencong). Pihak yang menyukai tab berargumen bahwa tab justru memberikan kebebasan bagi si pelihat teks untuk mengeset lebar indentasi. Tab juga mengirit ukuran file karena 1 karakter tab dapat menggantikan beberapa spasi.

Lebar tab default di program-program Unix adalah 8 karakter. Tapi setiap ditemukan tab bukan berarti "cetak 8 karakter spasi", melainkan "cetak spasi *hingga ke kolom kelipatan 8 berikutnya*." Misalnyaalnya pada teks "abc\tdef", maka \t di sini akan dicetak sebagai 5 spasi sehingga "d" akan mulai di kolom 9. Sementara pada teks "abcdefg\thij", maka \t di sini hanya akan dicetak sebagai 1 spasi sehingga "h" juga akan mulai kolom 9.

Karena itu prinsip mengubah tab ke/dari spasi adalah menghitung posisi kolom karakter demi karakter.

tab2space.pl. Inti program adalah baris 14-18, yang akan mengganti setiap karakter tab dengan sejumlah spasi. Berapa jumlah spasi yang diperlukan? Dihitung dengan rumus di baris 16, yaitu selisih antara lebar kolom saat ini (`length($1)`, semua karakter nontab dari kolom 1) dan posisi kolom kelipatan 8 berikutnya.

space2tab.pl. Untuk mengkonversi spasi ke tab sedikit lebih repot. Lihat baris 17. Pertama-tama kita mengambil token sebuah tab (`/\t/`) atau deretan spasi (`/ +/`) atau deretan nonspasi (`/[^ ]+/`). Jika ditemukan tab (baris 20-22) kita membiarkannya tetap menjadi tab. Demikian pula jika ditemukan deretan nonspasi (23-25). Tapi jika kita menemukan deretan spasi, kita harus mengganti dulu sebanyak mungkin spasi yang ada dengan tab (baris 30-42). Di baris 45-49 kita memilih tidak mengganti deretan spasi dengan tab jika tidak terjadi pengiritan jumlah karakter (misalnyaalnya: dua spasi dibandingkan satu tab + satu spasi; maka yang dipilih adalah dua spasi saja).
Kedua program Perl ini dapat menerima opsi `-w` untuk mengeset lebar tab. Defaultnya adalah 8 karakter. Saya sendiri biasanya mengeset tab menjadi 4 karakter di editor teks saya.

Catatan: di Unix terdapat perintah **expand** dan **unexpand** untuk melakukan hal yang sama (expand mengubah tab menjadi spasi; unexpand sebaliknya). Kedua perintah ini memiliki beberapa opsi tambahan, misalnyaalnya hanya mengubah tab/spasi di awal baris (indentasi) saja, dan mengizinkan posisi setiap tab diset custom (tidak hanya selalu kelipatan 8 kolom, tapi setiap tab dapat mengeset ke posisi berbeda-beda ukuran).

2-10 Mencari/Mengganti Daftar Kata

Masalah

Anda ingin mencari/mengganti kata-kata yang terdapat dalam daftar tertentu, misalnyaalnya kata-kata pada kamus. Misalnyaalnya Anda ingin memfilter/menyensor kata-kata makian atau kata-kata kasar.

Contoh kode PHP

```
 1|<?
  |
 3|//
 4|// sensor-kata.php
 5|//
  |
 7|// daftar kata-kata kasar. buat pembaca: maaf :-)
 8|$bad_words = array(
 9|  "dick[ -]?head",
10|  "cock[ -]?suck(ers?|ing)",
11|  "(mother|motha)[ -]?(fucker|fuckah?)",
12|  "asu",
13|  "anj(is|rit)",
14|  // dan seterusnya.
15|);
  |
17|// bentuk regex
18|$re = "\b(" . join("|", $bad_words) . ")\b";
  |
  |
21|if (!isset($submit)) $submit = "";
  |
23|if ($submit) {
24|  echo preg_replace("/$re/i", "***", $teks);
25|} else {
```

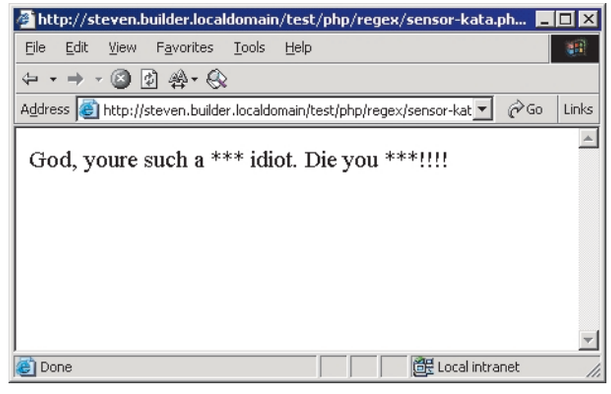


```
26|  echo "
27|    <form method=POST>
28|    <textarea name=teks cols=80 rows=6></textarea><br>
29|    <input type=submit name=submit>
30|    </form>
31|  ";
32|}
  |
34|?>
```

Contoh output program

Diskusi

Untuk mencari banyak kata dalam sebuah teks, ada beberapa alternatif pendekatan yang dapat dilakukan. **Pertama**, untuk setiap kata, mencarinya dengan fungsi string `pos()/strpos()`. Cara ini lambat jika jumlah katanya banyak dan teksnya panjang, karena program berulang kali melakukan full-scan pada teks untuk setiap kata.



Gambar 4-6.

Kedua, memecah teks menjadi array kata (misalnya: menggunakan regex) lalu mengetes setiap kata. Daftar kata yang ingin dicari disimpan dalam key array, misalnya:

```
$daftar_kata = array("kata1" => 1, "kata2" => 1, "kata3" => 1,
...)
```

Cara ini cepat dalam mengetes apakah sebuah kata terdapat dalam array kata, cukup dengan menggunakan fungsi PHP

`isset($daftar_kata[strtolower($kata)])`. Tapi cara ini kurang fleksibel, misalnya jika kita ingin mencari kata yang dapat dieja secara berbeda maka untuk tiap variasi ejaan kita memerlukan key baru di `$daftar_kata`.
Cara **ketiga** adalah dengan menggunakan regex yang berisi alternatif kata:

```
/\b(kata1|kata2|kata3|...)\b/i
```

Cara ini cukup cepat karena mesin regex sudah teroptimasi untuk mencari apakah salah satu dari kata-kata tersebut terdapat dalam teks. Tiap `kata1`, `kata2`, dan seterusnya. Juga dapat merupakan regex sehingga mempermudah kita menangkap perbedaan ejaan yang ada. Contoh kode di resep ini, `sensor-kata.php`, menggunakan cara ketiga. Pertama kita mendaftar kata-kata yang ingin disensor di array `$bad_words` (baris 8-15). Lalu kita menggabungkan tiap kata menjadi satu regex panjang `$re` (baris 18). Setelah itu tinggal lakukan pencocokan dengan regex ini. Di baris 24 kita mengganti setiap kata yang ingin disensor dengan ***. Anda juga bisa menggantinya dengan karakter * sejumlah panjang kata (misalnya: "bajingan" dengan 8 buah tanda "*") dengan cara berikut:

```
preg_replace("/$re/ie", "str_repeat('*', strlen('\\1'))", $teks);
```

Catatan: menyensor semua kata-kata kasar/ofensif bisa jadi amat sulit, karena pengunjung situs yang nakal bisa saja menggunakan segala macam variasi ejaan atau modifikasi pada teks untuk menghindari skrip sensor yang simplistik ini. Jika benar-benar ingin menghindari kata-kata ofensif terposting ke situs Anda, cara yang lebih efektif adalah dengan sistem moderasi—meskipun membutuhkan tangan manusia.

2-11 ROT-13

Masalah

Anda ingin mengacak pesan rahasia Anda sehingga tidak dapat dibaca orang lain.

Contoh kode Perl

```
 1|#!/usr/bin/perl
  |
 3|#
 4|# rot13.pl
 5|#
  |
 7|while (<>) {
 8|  s{([A-Za-z])}
 9|   { $1 ge "A" && $1 le "M" || $1 ge "a" && $1 le "m" ?
10|     chr(ord($1)+13) : chr(ord($1)-13) }eg;
11|  print;
12|}
```

Contoh keluaran program

```
$ echo 'Ini pesan rahasia, tidak dapat dibaca!' | ./rot13.pl
Vav crfna enunfvn, gvqnx qncng qvonpn!
```

Diskusi

ROT-13 adalah mekanisme penyandian teks di mana setiap huruf diputar 13 ke kanan (tapi bisa juga ke kiri). Misalnyaalnya, `a` (huruf ke-1 alfabet) menjadi `n` (huruf ke-14), `b` menjadi `o`, `s` (19) menjadi `f` (6), dan seterusnya. Kalau tidak salah ingat, katanya metode ini pernah dipakai Julius Caesar di zaman dahulu dalam menyampaikan pesan rahasia—meskipun saya amat meragukannya. Metode ini tentu saja sangat simpel dan tidak bisa diandalkan untuk enkripsi serius; ROT-13 dibahas di buku ini hanya demi demonstrasi regex.

Catatan: Di Perl sebetulnya ada cara lebih mudah untuk transformasi satu huruf, yaitu: `tr/A-Ma-mN-Zn-z/N-Zn-zA-Ma-m/`.

2-12 CSV

Masalah

Anda ingin memparse file CSV.

Pola regex

```
# A. pemisalnyaah field = koma, pengutip field = kutip ganda
/("(?:\\"|\\\\|[^",])*"|[^",]*)(?:,|\z)/g

# B. pemisalnyaah field = titik koma, pengutip field = kutip
tunggal
/('(?:\\'|\\\\|[^';])*'|[^';]*)(?:;|\z)/g
```

Contoh kode Perl

```
 1|#!/usr/bin/perl
  |
 3|#
```

```
 4|# parse-csv.pl
 5|#
  |
 7|# hasil parsing ada di array @records. setiap record adalah
  |  array field.
  |
 9|@records = ();
  |
11|while (<>) {
12|  # buang newline
13|  chomp;
  |
15|  my @fields = /("(?:\\"|\\\\|[^",])*"|[^",]*)(?:,|\z)/g;
16|  for (grep {/^"/} @fields) { s/^"//; s/"\z//; s/\\(.)/$1/g }
  |
18|  push @records, [@fields];
19|}
  |
21|# pakai @records ...
```

Diskusi

File `.CSV` adalah file yang berisi record. Setiap record terdiri dari satu atau lebih field. Antarrecord dipisahkan oleh newline. Antarfield dipisahkan oleh koma. Setiap field dapat dikutip dengan kutip ganda jika mengandung koma, agar koma tersebut tidak dianggap pemisalnyaah field. Field juga dapat mengandung `\"` yang berarti `"` literal. Field dapat mengandung `\\` yang berarti `\` literal.
Ada varian `.CSV` yang pemisalnyaah fieldnya adalah titik koma dan/atau pengutip fieldnya adalah kutip tunggal.

Contoh sebuah record `.CSV`:

```
"One","\"Two\"","Three\"","Four","\\","\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\"\\\\"
```

record ini berisi empat field, yaitu:

```
One
"Two"
Three"
Four
\
\\\\\\\\\\\"\\
```


Mari lihat pola regex secara lebih mendalam. Untuk setiap field, kita memberi dua pilihan. Pertama adalah field yang dikutip:

```
"(?:\\"|\\\\|[^",])*"
```

subpola ini berbunyi: string yang diawali " dan berisi 0 atau lebih `\\`, atau `\"`, atau karakter lain yang bukan pemisah field/pengutip field, dan diakhiri `"`. Pilihan kedua adalah field yang tidak dikutip:

```
[^",]*
```

Subpola ini berbunyi: 0 atau lebih karakter lain yang bukan pemisalnyaah field/pengutip field.

Lihat baris 16 program. Setelah kita mengambil semua field menggunakan `m//g`, kita mencari field yang dikutip dan melucuti kutipnya. Kita juga mengganti `\\` dengan `\`, dan `\"` dengan `"`.

Catatan: Karena memparse CSV adalah salah satu pekerjaan yang sering dilakukan, di Perl sudah disediakan modul `Text::CSV` dan `Text::CSV_XS`. Di PHP ada fungsi `fgetcsv()`.

Sebagai tantangan untuk Anda, Microsoft Excel dapat menghasilkan file CSV di mana sebuah field yang dikutip dapat mengandung newline, contohnya:

```
1,2,3,"Ini field yang
mengandung newline",5,6,7
```

Bagaimana cara memparse CSV seperti ini? Hint: Pertama, untuk mudahnya sebaiknya Anda tidak membaca isi file per baris, karena sebuah record dapat terdiri dari beberapa baris. Anda dapat mengambil seluruh isi file ke dalam sebuah string, lalu menggunakan pola seperti ini:

```
/("(?:\\"|\\\\|[^",])*"|[^",\012\015]*)(,|[\012\015]+)/sg
```

untuk mengambil seluruh field di seluruh record dalam bentuk array, berikut pemisalnyaah field/pemisalnyaah recordnya. Selanjutnya Anda tinggal mengiterasi array ini dan mencari pemisalnyaah record (`/[\012\015]/`) untuk membelah-belah deretan field menjadi beberapa record.

3-1 Mengganti Nama File

Masalah

Anda ingin mengganti nama beberapa/banyak file secara otomatis menggunakan pola tertentu, di mana hal ini sulit/repot jika dilakukan dengan perintah OS yang tersedia (seperti **mv** di Unix).

Contoh kode Perl

```
 1|#!/usr/bin/perl
  |
 3|#
 4|# perlrename.pl
 5|#
  |
 7|use Getopt::Std;
 8|getopts('de:hv');
  |
10|if ($opt_h) { print <<USAGE; exit 0 }
11|Rename files using Perl code.
  |
13|Usage: $0 <-e 'code'> file ...
  |
15|Options:
16| -e  Specify code to rename file (\$_), e.g. 's/\.old\$/
   \.bak/'
17| -v  Verbose
18| -d  Dry-run (implies -v)
19| -h  Show this help
20|USAGE
  |
22|die "FATAL: Code (-e) not specified, use -h for help\n"
   unless $opt_e;
23|eval $opt_e; die "FATAL: Code (-e) does not compile: $@\n" if
   $@;
  |
25|@files = ();
  |
27|if ($^O =~ /win32/i) {
28|  for (@ARGV) {
```



```
29|    if (/[*?]/) { push @files, glob $_ } else { push @files,
   $_ }
30|  }
31|} else {
32|  push @files, @ARGV;
33|}
  |
35|for (@files) {
36|  $OLD = $_;
37|  eval $opt_e;
38|  if ($opt_v || $opt_d) {
39|    print "$OLD -> $_\n";
40|  }
  |
42|  unless ($opt_d) {
43|    rename $OLD, $_;
44|    warn "ERROR: $OLD -> $_: $!\n" if $!;
45|  }
46|}
```

Diskusi

Seringkali kita ingin mengganti nama banyak file. Di Windows hal ini sulit dilakukan karena dengan Explorer kita harus mengganti satu per satu, sementara di DOS prompt perintah command-line yang disediakan Windows kemampuannya terbatas.

Di shell Unix untuk beberapa kasus kita dapat melakukan ini dengan looping, misalnyaalnya untuk mengganti file-file menjadi berekstensi `.txt`:

```
for f in README ChangeLog INSTALL AUTHORS; do mv $f $f.txt; done
```

Tapi untuk kasus-kasus yang lebih kompleks kita harus melakukan kombinasi dengan perintah `sed`, `test`, dan lain sebagainya. Belum lagi regex di perintah-perintah Unix kurang ampuh.

Solusinya adalah dengan menggunakan bahasa seperti Perl atau PHP. Program di resep ini, `perlrename.pl`, memperpraktis penggunaan Perl untuk merename file, sehingga kita tidak perlu membuat kode Perl setiap kali ingin merename sekelompok file, tapi cukup menyebutkannya di command line dengan opsi `-e`. Saya menggunakan skrip ini sehari-hari. Contoh, untuk mengubah nama file menjadi huruf kecil:

```
$ perlrename.pl -ve '$_ = lc($_)' *
bignum.c -> bignum.c
ChangeLog -> changelog
compar.c -> compar.c
COPYING -> copying
COPYING.ja -> copying.ja
GPL -> gpl
LEGAL -> legal
LGPL -> lgpl
Makefile.in -> makefile.in
MANIFEST -> manifest
README -> readme
README.ja -> readme.ja
ToDo -> todo
```

Kode Perl disebutkan di opsi `-e` ("eval") dan kode ini akan dijalankan untuk setiap file. Nama file disediakan di variabel `$_` dan kita bebas mengubah variabel ini menjadi nama baru. Opsi `-v` ("verbose") untuk menampilkan setiap file yang diganti namanya.

Beberapa contoh yang lebih kompleks (dan menggunakan regex tentunya) sebagai berikut. Saya memiliki file-file mailbox bernama `2002-10`, `2002-11`, dan lain sebagainya. Saya ingin mengubahnya menjadi `200210.mbox`, `200211.mbox`, dan lain sebagainya.

```
$ perlrename -e 's/-//; $_ .= ".mbox"' *
```

Seandainya saya ingin membalikkannya lagi (dari `200211.mbox` menjadi `2002-11`):

```
$ perlrename -e 's/(\d\d\d\d)(\d\d)\.mbox/$1-$2/' *.mbox
```

Saya ingin memindahkan file-file musik dan video (dengan berbagai ekstensi) ke subdirektori `tmp/`:

```
$ perlrename -e '$_ = "tmp/$_" if
  m#\.(mpe?g|avi|ogg|mp[234]|rm(vb?)|r[am]|asf|wmv)$/#i;' *
```

Mengubah ekstensi `.tgz` menjadi `.tar.gz`:

```
$ perlrename -e 's/\.tgz$/.tar.gz/' *.tgz
```


Saya memiliki file-file .mp3 yang diberi nama dengan pola `"00 - NAMAARTIS - JUDULTRACK.mp3"`. Saya ingin mengubahnya menjadi `"JUDULALBUM - 00 – JUDULTRACK.mp3"`:

```
$ perlrename -e 's{^(\d+) - .+ - (.+)\.mp3}
{Dangerously In Love – $1 - $2.mp3}' *.mp3
```

Harap berhati-hati menggunakan skrip ini karena jika kode Perl di -e salah, file-file Anda dapat tertimpa atau terganti namanya secara kacau. Gunakan dulu opsi `-d` ("dry-run", "mode coba-coba") yang saya sediakan agar file tidak betul-betul direname, melainkan skrip hanya menampilkan seperti apa perubahan yang akan dilakukan. Setelah dilihat kira-kira benar, barulah hilangkan opsi `-d`.

3-2 Wildcard

Masalah

Anda ingin mencocokkan nama file atau teks menggunakan sintaks wildcard.

Contoh kode Perl

```
  1|#!/usr/bin/perl
   |
  3|#
  4|# wildcard2re.pl
  5|#
   |
  7|sub wildcard2re {
  8|  my $wildcard = shift;
   |
 10|  my $state = {
 11|    in_single_quote => 0,
 12|    in_double_quote => 0,
 13|  };
   |
 15|  my $subst = sub {
 16|    my $token = shift;
   |
 18|    if ($token =~ /\\(.)/) {
 19|      if ($state->{in_single_quote}) {
 20|        if ($1 eq "'") {
 21|          $state->{in_single_quote} = !$state-
```

```
   >{in_double_quote};
22|          return quotemeta("\\");
23|        } else {
24|          return quotemeta($token);
25|        }
26|      } else {
27|        return quotemeta($1);
28|      }
29|    } elsif ($token eq '"') {
30|      if (!$state->{in_single_quote}) {
31|        $state->{in_double_quote} = !$state-
   >{in_double_quote};
32|        return "";
33|      } else {
34|        return '"';
35|      }
36|    } elsif ($token eq "'") {
37|      if (!$state->{in_double_quote}) {
38|        $state->{in_single_quote} = !$state-
   >{in_single_quote};
39|        return "";
40|      } else {
41|        return "'";
42|      }
43|    } elsif ($token eq '*') {
44|      if ($state->{in_single_quote} || $state-
   >{in_double_quote}) {
45|        return quotemeta("*");
46|      } else {
47|        return ".*";
48|      }
49|    } elsif ($token eq '?') {
50|      if ($state->{in_single_quote} || $state-
   >{in_double_quote}) {
51|        return quotemeta("?");
52|      } else {
53|        return ".";
54|      }
55|    } else {
56|      return quotemeta($token);
57|    }
```



```
58|  };
  |
60|  $wildcard =~ s/(\\.|'|"|\*|\?|[^\\'"*?]+)/$subst->($1)/egs;
  |
62|  '^' . $wildcard . '$';
63|}
  |
65|# test
66|for(qw/satu* "satu*" "s"atu* sa*tu? '"satu.*"'/, 'satu\*',
   "satu'\\*'") {
67|  printf "%s = %s\n", $_, wildcard2re($_);
68|}
```

Output program Perl

```
$ ./wildcard2re.pl
satu* = ^satu.*$
"satu*" = ^satu\*$
"s"atu* = ^satu.*$
sa*tu? = ^sa.*tu.$
'"satu.*"' = ^"satu\.\*"$
satu\* = ^satu\*$
satu'\*' = ^satu\\\*$
```

Diskusi

Anda mungkin ingin menyediakan fasilitas bagi user program Anda untuk mencari sebuah teks atau file secara fleksibel. Tapi sintaks regex mungkin terlalu teknis atau "geeky" bagi user Anda. Banyak user yang sudah mengerti sintaks wildcard DOS/Windows, yang hanya memiliki karakter meta * dan ?. Program `wildcard2re.pl` di atas berisi sebuah fungsi yang menerima pola wildcard dan menghasilkan pola regex yang ekivalen. Dengan demikian, Anda bisa mencari file atau teks di array atau di variabel mana saja dengan fungsi-fungsi regex biasa. Sementara masukan dari user Anda berupa pola wildcard yang lebih sederhana.

Beberapa perbedaan sintaks wildcard dibanding regex adalah sebagai berikut. **Pertama**, * di wildcard sama dengan `.*` di regex (0 atau lebih karakter apa saja). **Kedua**, ? di wildcard sama dengan `.` di regex (1 karakter apa saja). **Ketiga**, pola wildcard secara default bersifat terjangkar, jadi sama dengan pola regex yang diapit oleh ^ dan $. Contohnya, `abc*def` di wildcard sebetulnya sama dengan `^abc.*def$` di regex.

Yang agak menyulitkan dalam mengkonversi wildcard ke regex adalah, di Unix * dan ? dapat dilindungi dengan tanda kutip (kutip tunggal atau kutip ganda) sehingga berarti literal. Atau dapat juga diescape dengan \ (misalnya: *) sehingga berarti literal. Jadi kita tidak bisa begitu saja melakukan `s/\*/.*/` untuk mengubah semua `"*"` menjadi `".*"`, karena kita harus mengecek keberadaan kutip dan backslash.

Ini adalah contoh problem parsing, di mana arti sebuah token bergantung pada konteks. Program `wildcard2re.pl` adalah contoh melakukan parsing dengan regex. Di baris 51 kita mendaftar semua token yang mungkin mengubah state. Token-token tersebut adalah:

```
\x    untuk mengescape x menjadi literal. catatan: \x sendiri
merupakan literal jika terdapat di dalam kutip tunggal.
'     untuk mengawali atau mengakhiri teks dalam kutip tunggal.
catatan: 'sendiri merupakan literal jika terdapat di dalam kutip
ganda.
"     untuk mengawali atau mengakhiri teks dalam kutip ganda.
catatan: " sendiri merupakan literal jika terdapat di dalam kutip
tunggal
*     untuk mewakili 0+ karakter. catatan: * merupakan literal
jika ada di dalam kutip atau diawali backslash.
?     untuk mewakili 0+ karakter. catatan: ? merupakan literal
jika ada di dalam kutip atau diawali backslash.
[^\\'"*?]+  karakter-karakter lain yang akan diinterpretasi
sebagai literal
```

Seperti dapat dilihat setiap token dapat mempengaruhi konteks dan mengubah interpretasi terhadap token lainnya. Misalnyaalnya, jika terdapat ' sebelum * maka * akan menjadi literal karena ada di dalam kutip tunggal. Sementara ' akan menjadi literal (tidak akan membuka konteks kutip) jika ada di dalam kutip lain (kutip ganda), dan seterusnya.

Untuk mencatat state, kita menggunakan hash (baris 10-13). Lalu untuk memroses tiap token dengan rutin di baris 18-58. Contohnya untuk memroses token ' kita mengecek dulu apakah kita berada dalam state `in_double_quote`. Jika ya, maka anggap ' sebagai karakter literal biasa. Jika tidak, kita membuka state `in_single_quote`.

Terakhir, setelah semua token kita ganti dengan padanan regexnya, kita menyisipkan jangkar pada pola regex hasil kita (baris 62).

Catatan: sebetulnya kita sebaiknya mengecek state akhir setelah baris 60, yaitu `in_double_quote` dan `in_single_quote` haruslah false. Jika salah satunya true, maka berarti pola wildcard input tidaklah dengan benar menutup kutip, misalnyaalnya: `abc"tes`.

3-3 File Biner

Masalah

Anda ingin mengetahui apakah sebuah file merupakan file teks atau file biner.

Pola regex

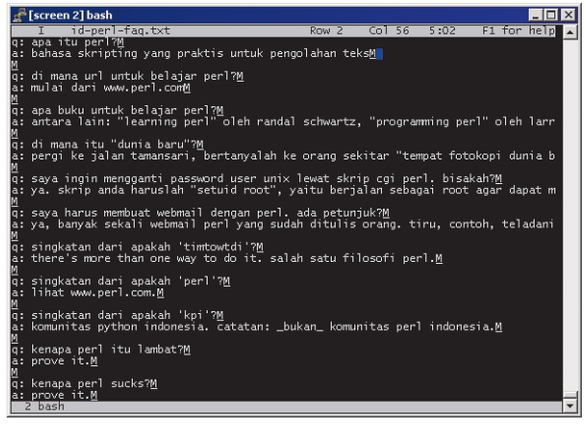
```
/[\000\200-\377]/
```

Contoh kode PHP

```
// catatan: file_get_contents() baru tersedia di PHP 4.3.0
function is_binary_file($path) {
  return preg_match('/[\000\200-\377]/',
file_get_contents($path)) ?
    true : false;
}
```

Diskusi

File biner adalah file yang mengandung karakter NUL (ASCII 0) atau karakter ASCII di atas 7 bit (ASCII 128-255). File teks boleh mengandung karakter cetak



Gambar 4-7.

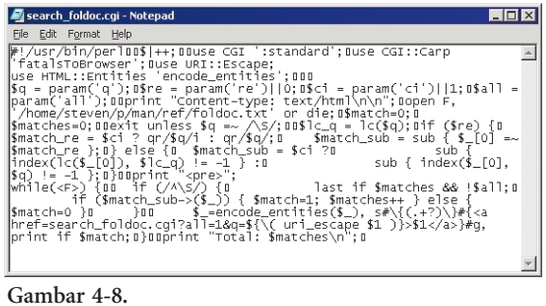
biasa (ASCII 32-127) atau karakter kontrol (1-31). Contoh karakter kontrol adalah tab (ASCII 9), escape (ASCII 27), dan lain sebagainya. Perhatikan pola \xxx di mana x adalah angka. Pola ini melambangkan karakter ASCII dalam basis oktal. 8 desimal = 010 oktal. 255 desimal = 377 oktal.

3-4 Konversi file DOS/Unix

Masalah

Anda memiliki file berformat DOS (di mana setiap baris diakhiri "\015\012") di Unix, sehingga di editor tertentu akan terjadi pemunculan huruf ^M yang mengganggu di setiap ujung baris file (lihat Gambar 4-7). Atau masalah lain, misalnya ketika Anda mengeksekusi file CGI, timbul kesalahan 500 Internal Server Error yang disebabkan perubahan baris `#!/usr/bin/perl` menjadi `#!/usr/bin/perl^M`.

Atau sebaliknya, Anda memiliki file teks Unix, (di mana setiap baris diakhiri "\012" saja) di DOS/Windows sehingga menimbulkan masalah, misalnya pada saat dibuka di teks editor Notepad, semua baris menjadi menyatu tanpa newline, yang muncul hanyalah karakter dengan bentuk kotak kecil (lihat Gambar 4-8).



Gambar 4-8.

Contoh kode Perl

```
# A. konversi Unix -> DOS, dilakukan di DOS/Windows
C:> perl -pi.bak -e "s/\012/\015\012/" UNIXFILE1.TXT ...

# B. konversi DOS -> Unix, dilakukan di Unix
$ perl -pi -e 's/\015//' DOSFILE1.TXT ...
```



```
# C. sama dengan A, tapi tak peduli apakah file sudah berformat DOS/Unix
C:> perl -pi.bak -e "s/\015*\012/\015\012/" UNIXORDOSFILE1.TXT ...

# D. sama dengan B, tapi tak peduli apakah file sudah berformat DOS/Unix
$ perl -pi -e 's/\015+/\012/' UNIXORDOSFILE1.TXT ...
Contoh kode Ruby
# A2. konversi Unix -> DOS, dilakukan di DOS/Windows
C:> ruby -pi.bak -e "$_.gsub /\012/, %Q(\015\012)" UNIXFILE1.TXT ...

# B2. konversi DOS -> Unix, dilakukan di Unix
$ ruby -pi -e '$_.gsub! /\015/, ""' DOSFILE1.TXT ...

# C2. sama dengan A2, tapi tak peduli apakah file sudah berformat DOS/Unix
C:> ruby -pi.bak -e "$_.gsub /\015*\012/, %Q(\015\012)" ^
   UNIXORDOSFILE1.TXT ...

# D2. sama dengan B2, tapi tak peduli apakah file sudah berformat DOS/Unix
$ ruby -pi -e '$_.gsub! /\015+/, ""' UNIXORDOSFILE1.TXT ...
```

Pembahasan

Kesalahan atau ketidakcocokan karakter newline di file teks biasanya ditimbulkan karena sata terjadi perpindahan file antara DOS dan Unix tidak dilakukan dalam mode ASCII FTP. Atau bisa jadi disebabkan karena saat perpindahan file antara DOS dan Unix dilakukan melalui protokol HTTP, Samba, SSH, dan lain sebagainya.

Untuk mengubah file dari DOS ke Unix atau sebaliknya, sebetulnya tidak diperlukan regex. Kita hanya cukup mengganti “\015\012” ke “\012” (atau sebaliknya jika dari Unix ke DOS). Namun dengan bantuan regex kita bisa lebih fleksibel. Lihat kode C misalnya.

Tak peduli apakah file sudah dalam format DOS atau masih Unix, kode C dapat mengubahnya menjadi format DOS yang benar. Sementara itu, jika Anda menggunakan kode A pada file DOS misalnya, akan menghasilkan file dengan akhiran “\015\015\012” yang dapat menimbulkan masalah.

3-5 Nomor Baris

Masalah

Anda ingin memberi nomor baris pada listing file.

Contoh kode Perl

```
  1|#!/usr/bin/perl
   |
  3|while (<>) {
  4|  if (/\S/) {
  5|    printf "%4d|", $.;
  6|  } else {
  7|    print  "    |";
  8|  }
  9|  print;
 10|}
```

Contoh output program Perl

```
$ cat linenum.pl
#!/usr/bin/perl

while (<>) {
  if (/\S/) {
    printf "%4d|", $.;
  } else {
    print  "    |";
  }
  print;
}
$ ./linenum.pl linenum.pl
   1|#!/usr/bin/perl
    |
   3|while (<>) {
   4|  if (/\S/) {
   5|    printf "%4d|", $.;
   6|  } else {
   7|    print  "    |";
   8|  }
   9|  print;
```



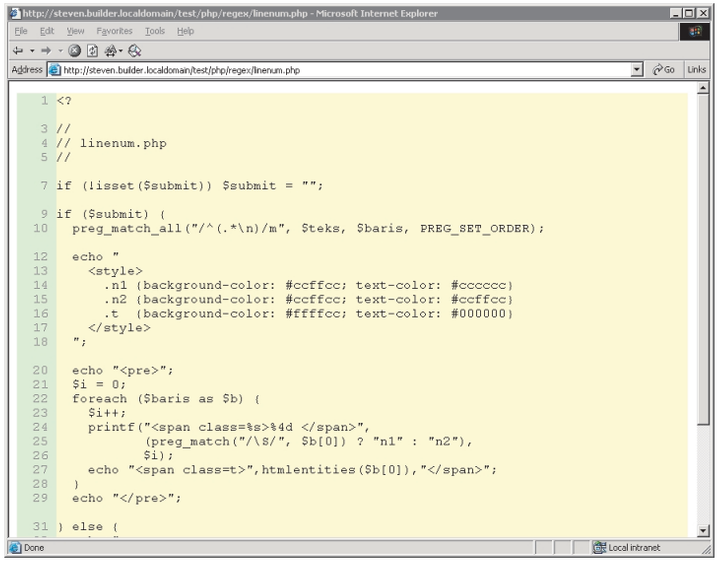
```
  10|}
$ ./linenum.pl linenum.pl > linenum.pl.txt
```

Contoh kode PHP

```
  1|<?
   |
  3|//
  4|// linenum.php
  5|//
   |
  7|if (!isset($submit)) $submit = "";
   |
  9|if ($submit) {
 10|  $teks = stripslashes($teks);
 11|  preg_match_all("/^(.*\n)/m", $teks, $baris,
    PREG_SET_ORDER);
   |
 13|  echo "
 14|    <style>
 15|      .n1 {background-color: #ccffcc; color: #999999}
 16|      .n2 {background-color: #ccffcc; color: #ccffcc}
 17|      .t  {background-color: #ffffcc; color: #000000}
 18|    </style>
 19|  ";
   |
 21|  echo "<pre>";
 22|  $i = 0;
 23|  foreach ($baris as $b) {
 24|    $i++;
 25|    $b[0] = preg_replace("/[\012\015]+/", '', $b[0]);
 26|    printf("<span class=%s>%4d </span>",
 27|           (preg_match("/\S/", $b[0]) ? "n1" : "n2"),
 28|           $i);
 29|    $b[0] .= str_repeat(" ", 80-strlen($b[0]));
 30|    echo "<span class=t>",htmlentities($b[0]),"</span>\n";
 31|  }
 32|  echo "</pre>";
   |
 34|} else {
 35|  echo "
```

```
    <form method=POST>
    <textarea name=teks cols=80 rows=6></textarea><br>
    <input type=submit name=submit>
    </form>
  ";
}
  |
?>
```

Contoh output program PHP



Gambar 4-9.

Diskusi

Kode Perl di resep ini adalah kode yang saya gunakan untuk membuat listing nomor baris di buku ini. Konsepnya sangat sederhana, cukup memberi prefiks nomor baris di depan setiap baris teks, namun jika baris tidak mengandung tulisan (tidak cocok dengan regex /\S/), maka jangan cetak nomor barisnya agar tampilan sedikit lebih bersih.

Di versi PHP nomor baris untuk baris kosong tetap dicetak, tapi diberi warna teks yang sama dengan warna latar belakang sehingga efeknya sama yaitu tak terlihat (lihat Gambar 4-9).

3-6 Menghapus File Backup

Masalah

Anda ingin membersihkan berbagai file backup di bawah direktori home Anda.

Pola regex

```
# A. editor 'joe' di Unix membuat backup dengan akhiran ~
/~$/

# B. editor 'PFE' di Windows membuat backup dengan pola *.$$$
/\.\$\$\$$/

# C. editor 'emacs' membuat backup dengan pola #FILE#
/^#.+#$/

# D. ekstensi backup lain yang umum
/\.(bak|bk!|!!!)$/i
```

Contoh kode Perl

```
use File::Find;

find sub {
  return unless -f;
  do { print "$_\n"; unlink $_ } if /~$/ or
                                    /\.\$\$\$$/ or
                                    /^#.+#$/ or
                                    /\.(bak|bk!|!!!)$/i;
}, ".";
```

Diskusi

Kode Perl di resep ini dapat menghapus file backup secara rekursif (file di semua subdirektori). Hati-hati menggunakannya, sebab bisa jadi Anda membuat backup file penting tapi memberinya nama `*.bak`.

3-7 Mengubah Tanggal File Sesuai Backup-nya

Masalah

Anda mungkin telah mengedit sebuah file, atau inplace-edit dengan skrip Perl atau Ruby (misalnyaalnya: dengan `perl -pi~ -e's/\015//' FILE.TXT` untuk

mengganti format DOS menjadi Unix; lihat resep 3-4). Namun akibatnya, tanggal file menjadi berubah. Anda ingin tanggal file yang semula.

Contoh kode Perl

```
# A. jika akhiran backup adalah ~
perl -e'for(@ARGV){utime((stat "$_~")[8,9],$_) unless /~$/ or !-
f}' *

# B. jika akhiran backup adalah .bak
perl -e'for(@ARGV){utime((stat "$_.bak")[8,9],$_) unless /\.bak$/
or !-f}' *
```

Diskusi

Silakan baca tentang fungsi `utime()` dan `stat()` di manual perl.

4-1 Mengescape Argumen Shell

Masalah

Anda ingin melewatkan isi sebuah string sebagai argumen perintah shell Unix, tapi Anda perlu memastikan bahwa karakter-karakter tak aman seperti kutip tunggal (') dan pipe (|) tidak membahayakan atau membuat jadi salah.

Contoh kode Perl

```
sub escapeshellarg {
  local $_ = shift;
  s/'/'"'"'/g;
  "'$_'";
}
```

Diskusi

Mengapa escaping perlu? Jika string `$str` berisi `"hello|world"` dan Anda ingin mencetaknya menggunakan `echo` Unix dengan perintah `system("echo $str");`. yang terjadi bukanlah string `hello|world` yang tercetak, namun hanya, hello. Shell mencoba menjalankan perintah world yang defaultnya tidak ada, sehingga muncul kesalahan "command not found"), karena setelah substitusi $str, perintah yang dikirim adalah:

```
echo hello|world
```


Bayangkan jika `$str` berisi `"hello|rm -rf /"`. Mengapit $str dengan tanda kutip tunggal saja masih belum cukup, karena `$str` sendiri masih mungkin mengandung kutip tunggal. Karena itu, yang dilakukan fungsi `escapeshellarg()` seperti di contoh kode di atas, adalah mengubah dulu kutip tunggal yang ada menjadi di luar kutip tunggal dan di dalam kutip ganda. Kutip ganda boleh mengapit kutip tunggal.

Catatan: PHP sudah memiliki fungsi seperti ini yaitu `escapeshellarg()`.

4-2 Memblokir Host

Masalah

Komputer Anda diserang oleh DDOS attack atau oleh worm seperti NIMDA atau oleh bot yang membombardir webserver Anda dengan request bertubi-tubi. Anda ingin memblokir sementara host-host yang melakukan serangan ini.

Contoh kode Perl

```
# A. memblokir NIMDA
$ perl -ane'
  BEGIN {
    %self_ips = map {$_=>1}
                ('/sbin/ifconfig' =~ /inet
addr:(\d+\.\d+\.\d+\.\d+)/g);
  }
  system "/sbin/iptables -A INPUT -s $F[0] -j DROP" if
    m#/winnt/system32# && !$seen{$F[0]}++ && !exists
$self_ips{$F[0]};
' /var/log/httpd/access.log

# B. memblokir sebuah DDOS attack
$ perl -ne'
  BEGIN {
    %self_ips = map {$_=>1}
                ('/sbin/ifconfig' =~ /inet
addr:(\d+\.\d+\.\d+\.\d+)/g);
  }
  system "/sbin/iptables -A INPUT -s $1 -j DROP" if
    /client (\d+\.\d+\.\d+\.\d+)\] request failed: erroneous / &&
    !$seen{$1}++ && !exists $self_ips{$1};
' /var/log/httpd/error.log
```

Diskusi

Perl dan Ruby menurut saya lebih ideal untuk pekerjaan seperti ini ketimbang PHP, dikarenakan fasilitasnya seperti opsi-opsi command line sehingga skrip kita dapat ditulis dan diedit langsung di shell tanpa harus membuat file skripnya dulu.

Opsi `-n` misalnya, berarti kode `-e` kita akan dibungkus dengan loop yang akan membaca masukan baris demi baris; skrip kita tinggal membaca tiap baris di variabel `$_`. Opsi `-a` berarti selain dibaca ke dalam `$_`, tiap baris masukan juga displit dengan regex ke dalam array `@F`. Kita tinggal membaca tiap field dengan `$F[0]`, `$F[1]`, dan lain sebagainya.

Contoh A adalah skrip yang pernah saya pakai untuk memblokir host yang terinfeksi NIMDA, worm yang pernah popular beberapa tahun lalu. Worm ini menginfeksi mesin Windows dan mencoba mencari mesin-mesin lain lewat lubang di port 80. Tentu saja mesin Unix/Linux tidak dapat diinfeksi webservernya, namun banjirnya request dari host yang terinfeksi NIMDA amat mengganggu karena mengotori log webserver. Skrip mula-mula mendaftar IP diri sendiri dengan mengambil IP dari perintah `/sbin/ifconfig`. Ini berguna agar kita tidak secara tak sengaja memblokir IP sendiri!

Setelah itu program akan mengecek baris demi baris file log webserver `/var/log/httpd/access.log`. Format `access.log` adalah sebagai berikut (semua dalam satu baris):

```
192.168.1.2 - - [01/Jan/2003:00:38:22 +0700] "GET /..." 200 3843
"http://steven.builder.localdomain/..." "Mozilla/4.0 (compatible;
MSIE 6.0; Windows NT 5.0; .NET CLR 1.0.3705)"
```

Field pertama (`$F[0]`) adalah IP/host pengakses. Jika Apache diset dengan `HostNameLookups On`, maka Apache akan mencoba meresolve IP ini. Kita anggap `HostNameLookups` adalah `Off`, sehingga field pertama selalu berisi alamat IP.

Skrip lalu mengecek apakah ada request berisi string `/winnt/system32`. Kita dapat cukup yakin bahwa di situs kita tidak ada file seperti itu, sementara string seperti itulah yang terdapat dalam request yang dikirim NIMDA. Karena itu kita putuskan bahwa string tersebut merupakan indikasi bahwa host terinfeksi NIMDA.

Sebelum memblokir si IP dengan perintah `/sbin/iptables`, kita mengecek dulu apakah `$seen{$F[0]}++` bernilai 0 (jika tidak 0, berarti host sudah pernah diblokir dan tidak perlu diblokir 2 kali). Kita juga mengecek apakah IP merupakan IP diri sendiri, `exists $self_ips{$F[0]}`. Jika ya, jangan blokir.

Contoh B mirip dengan contoh A, perbedaannya hanyalah kita mengecek `error.log` dan bukan `access.log`, serta pola yang dicari berbeda. Beberapa waktu lalu salah satu server yang saya urus pernah diserang DDOS attack yang mengirimkan request garbage ke port 80. Apache melaporkan hal ini di error log dengan pesan seperti sebagai berikut:

```
[localhost] [Fri Dec  5 10:00:55 2003] [error] [client
12.34.56.78] request has failed: erroneous characters after
protocol string: :QNDMOSB!w@...
```

Tentu saja dengan memblokir menggunakan `/sbin/iptables` saja DDOS tetap berlanjut dan menguras bandwidth, namun setidaknya webserver Apache terbebas dari banjir request host-host ini dan dapat terus berjalan.

Resep ini dapat dengan mudah Anda modifikasi untuk memblokir robot, user yang nakal, dan lain sebagainya; prinsipnya hanyalah melihat file log yang sesuai (apakah lastlog, log FTP/xferlog, syslog /var/log/messages, maillog, dan lain sebagainya), mencari pola yang sesuai, dan mengekstrak alamat IP/host dari log.

4-3 Crontab Seram

Masalah

Anda ingin mengetahui user mana di sistem Unix/Linux Anda yang memasang crontab dengan entri yang dijalankan lebih sering dari 10 menit sekali.

Contoh kode Perl

```
# cd /var/spool/cron
# for file in *; do
    perl -lne'
      s/^\s+//;
      next if /^#/;
      next unless @f=split /\s+/,$_,6;
      print "($ARGV:$.): $_"
      if $f[0]=~m#\*(/[1-9])?\b|(\d+,){5,}#;' $file
  done
```

Diskusi

Crontab semua user disimpan di `/var/spool/cron/<NAMAUSER>`, di mana `<NAMAUSER>` adalah nama user Unix (`root`, `steven`, dan lain sebagainya). Skrip

pada contoh menganalisis semua file crontab baris demi baris. Pertama-tama kita melakukan "ltrim" dengan regex `s/^\s+//`. Lalu kita melewati baris komentar. Lalu kita mencoba membagi menjadi 6 field dengan `split()` sebagaimana format crontab. Yang menjadi perhatian kita adalah field pertama yaitu menit.

Setiap field waktu di entri crontab dapat berbentuk:

```
\d+
\d+(,\d+)+
\*
\*/\d+
```

Contohnya, untuk field menit (`$f[0]` = field pertama):

```
3 = di menit ketiga
3,10,15 = di menit ketiga, sepuluh, dan lima belas
* = setiap menit (!)
*/5 = setiap lima menit sekali
```

Jadi untuk mencari mana entri yang dijalankan lebih sering dari 10 menit sekali, kita mencoba mencari pola `*/1`, `*/2`, hingga `*/9`. Atau mencari deretan angka minimal 6 buah (misalnya: 1,11,14,20,45,55; di mana ada 2 buah item yang terjadi kurang dari 10 menit sekali yaitu dari menit 11 ke 14 dan dari 55 ke 1).

4-4 User yang Tidak Pernah Login

Masalah

Anda ingin mengetahui user-user mana di mesin Unix Anda yang tidak pernah login (baik FTP maupun shell).

Contoh kode Perl

```
# perl -le'
    open P,"/etc/passwd";while(<P>){/([^:]+):[^:]+:(\d+):/ and
$u{$1}=$2}
    open L,"last|";while(<L>){/(\S+)/;delete $u{$1}}
    for(sort keys %u){print "$_ (uid $u{$_})" if $u{$_} &&
$u{$_}>=500}'
```

Diskusi

Pertama skrip membaca daftar user di sistem dengan memparse file `/etc/passwd`.

Format file ini sederhana:

```
steven:x:500:500:steven:/home/steven:/bin/bash
```

yaitu setiap record terdiri satu baris dan setiap field dipisahkan tanda titik dua (`:`). Field pertama adalah username, field ketiga UID, dan dua itu yang kita ambil dan kita masukkan ke hash `%u`.
Skrip lalu membaca output perintah **last** untuk mengetahui siapa yang terakhir login ke mesin. Output stat adalah seperti ini:

```
steven   pts/5                        Fri Jan  9 16:23   still
logged in
steven   pts/3                        Fri Jan  9 15:58   still
logged in
steven   pts/8                        Fri Jan  9 15:57 - 15:57
(00:00)
steven   pts/9                        Fri Jan  9 15:54 - 15:57
(00:02)
```

sehingga kita cukup mengambil field pertama untuk mengambil username. Setiap username yang terdapat di last berarti pernah login, dan kita menghapusnya dari hash `%u`. Langkah selanjutnya adalah melihat sisa user biasa (UID >= 500) yang masih tersisa di hash `%u`. Inilah user-user yang tidak pernah login. Kita perlu melakukan pengecekan UID minimum karena banyak user sistem (seperti `nobody`, `ftp`, `apache`, `bin`, dan lain sebagainya) juga tidak pernah "login".

4-5 Frekuensi Login User

Masalah

Anda ingin mengetahui user mana yang paling sering login ke mesin Unix Anda.

Contoh kode Perl

```
# perl -le'
    open P,"/etc/passwd";while(<P>){/([^:]+):[^:]+:(\d+):/
;$u{$1}=[$2,0]}
    open L,"last|";while(<L>){/(\S+)/;$u{$1}[1]++}
    for(sort {$u{$b}[1]<=>$u{$a}[1] or $a cmp $b} keys %u){
      print"$_ (uid $u{$_}[0]) = $u{$_}[1]" if
$u{$_}[0]&&$u{$_}[0]>=500
    }'
```

Contoh output program

```
steven (uid 500) = 145
adserver (uid 526) = 30
builder (uid 530) = 4
mirrors (uid 1001) = 0
mwn (uid 529) = 0
nfsnobody (uid 65534) = 0
testsuite (uid 525) = 0
tomi (uid 531) = 0
```

Diskusi

Resep ini merupakan varian resep sebelumnya, 4-4. Perlu dicatat juga bahwa sebetulnya jika username terlalu panjang, akan terpotong waktu di output **last**.

Perhatikan juga bahwa defaultnya `/var/log/wtmp` (yang mencatat login tiap user) dirotate atau direset secara berkala.

4-6 Mengecek Integritas Paket RPM

Masalah

Anda ingin mengetahui program-program mana yang sudah diganti, karena server Anda tersusupi orang dan program-programnya (misalnya: `/bin/login` atau `/usr/bin/passwd`) disuntiki trojan. Atau, rekan admin yang mengganti program tertentu.

Contoh kode Perl

```
# A. tidak mencetak nama paket tempat program berada
$ rpm -Va | perl -ne 'print if m#/(s?bin|lib)/#'

# B. menyebutkan nama paket tempat program berada
$ perl -e 'open A,"rpm -qa|";
    while ($pkg=<A>) {
      chomp $pkg;
      open V,"rpm -V $pkg|";
      while (<V>) { chomp; print "$_ ($pkg)\n" if m#/(s?bin|lib)#
}
    }'
```

Contoh output program B


```
.M.....   /usr/lib/courier-imap/libexec/authlib/
authdaemond.pgsql (courier-imap-pgsql-2.0.0-1.7.3)
misalnyasing    /usr/X11R6/lib/X11/fonts/100dpi/encodings.dir
(XFree86-base-fonts-4.2.0-8)
misalnyasing    /usr/X11R6/lib/X11/fonts/75dpi/encodings.dir
(XFree86-base-fonts-4.2.0-8)
misalnyasing    /usr/X11R6/lib/X11/fonts/CID/encodings.dir
(XFree86-base-fonts-4.2.0-8)
misalnyasing    /usr/X11R6/lib/X11/fonts/CID/fonts.dir (XFree86-
base-fonts-4.2.0-8)
misalnyasing    /usr/X11R6/lib/X11/fonts/CID/fonts.scale
(XFree86-base-fonts-4.2.0-8)
misalnyasing    /usr/X11R6/lib/X11/fonts/Speedo/encodings.dir
(XFree86-base-fonts-4.2.0-8)
misalnyasing    /usr/X11R6/lib/X11/fonts/Type1/encodings.dir
(XFree86-base-fonts-4.2.0-8)
misalnyasing    /usr/X11R6/lib/X11/fonts/local/encodings.dir
(XFree86-base-fonts-4.2.0-8)
misalnyasing    /usr/X11R6/lib/X11/fonts/local/fonts.dir
(XFree86-base-fonts-4.2.0-8)
misalnyasing    /usr/X11R6/lib/X11/fonts/misalnyac/encodings.dir
(XFree86-base-fonts-4.2.0-8)
.....UG. c /var/lib/rpm/__db.001 (rpm-4.0.4-7x.18)
.....UG. c /var/lib/rpm/__db.002 (rpm-4.0.4-7x.18)
.M.....   /usr/bin/rblsmtpd (ucspi-tcp-0.88-1)
....L...   /usr/bin/vauthenticate (vmailmgr-0.96.9-2)
.M.....   /usr/lib/courier-imap/libexec/authlib/authdaemon
(courier-imap-2.0.0-1.7.3)
.M.....   /usr/lib/courier-imap/libexec/authlib/authdaemond
(courier-imap-2.0.0-1.7.3)
.M.....   /usr/lib/courier-imap/libexec/authlib/
authdaemond.plain (courier-imap-2.0.0-1.7.3)
.M.....   /usr/lib/courier-imap/libexec/authlib/
authdaemond.mysql (courier-imap-mysql-2.0.0-1.7.3)
....L...   /usr/lib/libfbclient.so.1 (firebird-client-embedded-
1.5.0.4027-4)
.M....G.   /home/sloki/cpanel/sbin (slokihf-1.1-4slk_shf11_rh72)
.M.....   /usr/lib/courier-imap/libexec/authlib/authdaemond.ldap
(courier-imap-ldap-2.0.0-1.7.3)
....L...   /usr/lib/libglide3.so.3 (Glide3-20010520-13)
```

Diskusi

Setiap paket RPM mencatat tanggal, ownership, permisalnyasion dan MD5 checksum file-filenya. Data ini tersimpan di database RPM (di `/var/lib/rpm/`). Kita dapat meng-query database ini melalui program **rpm** dan opsi `-V` (verify). Contoh output `rpm -V`:

```
$ rpm -V samba-client
.M...... /usr/bin/smbmount
.M...... /usr/bin/smbumount
```

artinya kedua program telah berubah permisalnyasionnya (M=mode) dibandingkan kondisi original di paket RPM. Contoh lain:

```
$ rpm -V samba-common
S.5....T c /etc/samba/lmhosts
S.5....T c /etc/samba/smb.conf
```

artinya kedua file konfigurasi telah berubah ukurannya (S=size), MD5 checksumnya (5=MD5), dan tanggal modifikasinya (T=time). Tidak heran, sebab file tersebut telah diedit.

Berubahnya MD5 checksum pada program binary seperti `/bin/login` merupakan salah satu pertanda kuat bahwa program tersebut telah diganti dengan trojan. Flag lain `rpm -V` bisa dilihat di dokumentasi rpm.

Catatan: Resep ini bukanlah solusi tercepat untuk meng-query database RPM. Anda bisa juga mengakses API RPM langsung via C atau Perl/Python.

4-7 Memparse File .INI

Masalah

Anda ingin memparse/mengekstrak informasi dari file .INI.

Pola regex

```
# A. untuk mencocokkan header section
/^\[(.+?)\]/

# B. untuk mencocokkan isi section
/^(.+?)=(.*)/
```

Contoh kode Perl


```
 1|#!/usr/bin/perl
  |
 3|#
 4|# parse-ini.pl
 5|#
  |
 7|# hasil parsing ada di hash %sections. setiap section adalah
  hash berisi
 8|# daftar key => value.
  |
10|%sections = ();
11|$current_section = "";
  |
13|while (<>) {
14|  # buang newline
15|  chomp;
  |
17|  # abaikan baris kosong
18|  next unless /\S/;
  |
20|  # abaikan komentar
21|  next if /^;/;
  |
23|  # header section
24|  if (/^\[(.+?)\]/) {
25|    $current_section = $1;
26|  }
  |
28|  # key=value
29|  elsif (/^(.+?)=(.*)/) {
30|    $sections{$current_section}{$1} = $2;
31|  }
  |
33|  # syntax error?
34|  else {
35|    warn "SYNTAX ERROR ON LINE $.: $_\n";
36|  }
  |
38|}
  |
40|# pakai %sections ...
```

Diskusi

.INI adalah format file konfigurasi berbasis teks yang mulai dikenal sejak Windows 3.x dan banyak dipakai oleh program-program Microsoft dan program Windows lainnya. Formatnya amat sederhana dan mudah dibaca. Berikut sebuah contohnya:

```
[Config]
WordWrap=0
HighlightURLs=0
AutoIndent=1
FileFormat=1
TabSize=8
Font=Lucida Console
dwEffects=0
FontColor=0
CharSet=0
BackColor=16777215
FontSize=160

[View]
Statusbar=1
Toolbar=1
SavePosition=0
RecentFiles=1
SelectionBar=1

[MRU]
MRU_FILE1=z:\buku\2\raw\examples.txt
MRU_FILE2=C:\tmp\20040109.txt
MRU_FILE3=C:\tmp\win32pad.ini
MRU_FILE4=C:\WINNT\system32\drivers\etc\hosts
MRU_FILE5=Z:\www\test\php\regex\nomor-cantik.php
```

Sebuah file .INI dapat terdiri dari beberapa section. Di mana tiap section dimulai dengan header `[NAMASECTION]`. Isi section adalah baris-baris dalam bentuk `KEY=VALUE`. Komentar dimulai dengan karakter titik koma (`;`). Semuanya terkumpul dalam satu file.

Sayangnya file .INI semakin jarang dipakai program-program Windows "modern", karena rata-rata semakin mengandalkan registry atau file XML. Padahal format .INI amat sederhana, mudah dibaca dan diedit dengan program editor teks manapun.

4-8 Melarang Password yang Mudah Ditebak

Masalah

Anda tidak ingin user Anda menggunakan password-password yang terlalu mudah atau simplistik, misalnyaalnya: "123" atau "abcd".

Contoh kode Perl

```
 1|#!/usr/bin/perl
  |
 3|#
 4|# bad-password.pl
 5|#
  |
 7|die "Usage: $0 <password>\n" unless @ARGV;
  |
 9|%kamus = ();
10|open F, "/usr/share/dict/words";
11|while (<F>) { chomp; $kamus{lc $_}++ }
  |
13|$bad_reason = bad_password($ARGV[0]);
14|if ($bad_reason) {
15|  print "Bad password: $bad_reason.\n";
16|} else {
17|  print "Password OK.\n";
18|}
  |
20|sub bad_password {
21|  local $_ = shift;
22|  my $lc = lc $_;
23|  my @huruf = split //, $lc;
24|  my %huruf = map {$_=>1} @huruf;
25|  my $i;
26|  my $ok;
  |
28|  return "terlalu pendek" if length($_) <= 5;
  |
30|  return "ada dalam kamus" if exists $kamus{lc($_)};
  |
32|  return "berbentuk seperti tanggal" if
33|    m#^\d{3}
```

```
34|       ([\s/-])
35|       (\d{2}|jan|[fp]eb|mar|apr|may|mei|ju[nl]|
36|        aug|agu|sep|o[kc]t|no[vp]|de[cs])
37|       \1
38|       \d\d(\d\d)?$#ix;
  |
40|  return "berbentuk seperti nomor telepon" if
41|    /^\d\d\d\d?[- ]\d\d\d\d$/;
  |
43|  return "berbentuk seperti nomor polisi kendaraan" if
44|    /^[A-Z]{1,2}[ -]\d{1,4}[- ][A-Z]{2}$/i;
  |
46|  $ok = 0;
47|  for ($i=0; $i < $#huruf; $i++) {
48|    my $diff = ord($huruf[$i+1]) - ord($huruf[$i]);
49|    $ok = 1 if $diff > 1 || $diff < -1;
50|  }
51|  return "merupakan urutan huruf/angka" if !$ok;
  |
53|  return "mengandung terlalu sedikit karakter berbeda"
54|    if keys %huruf <= 3;
  |
56|  my $foo = $lc;
57|  $foo =~ y/qwertyuiop[]asdfghjkl;'zxcvbnm,.\//a-la-ka-j/;
58|  $foo =~ y/!@#\$%^&*()_+|~/abcdefghijklmn/;
59|  $foo =~ y/-1234567890=\\'/kabcdefghijlmn/;
60|  $ok = 0;
61|  my @foo = split //, $foo;
62|  for ($i=0; $i < $#foo; $i++) {
63|   my $diff = ord($foo[$i+1]) - ord($foo[$i]);
64|   $ok = 1 if $diff > 1 || $diff < -1;
65|  }
66|  return "mengandung huruf-huruf bersebelahan di keyboard" if
   !$ok;
  |
68|  # pola ababab, abcabc, abcdabcd
69|  return "mengandung pengulangan $1" if
70|    $lc =~ /^(..)\1\1/ or
71|    $lc =~ /^(...)\1/ or
72|    $lc =~ /^(....)\1/;
  |
```



```
74|  # pola abccba, abcddcba
75|  return "mengandung pengulangan $1" if
76|    $lc =~ /^(.)(.)(.)\3\2\1/ or
77|    $lc =~ /^(.)(.)(.)(.)\4\3\2\1/;
  |
79|  # password ok
80|  return "";
81|}
```

Diskusi

Masalah ini mirip dengan resep 1-4 Nomor Cantik, yaitu mencari keteraturan pada sebuah string; hanya saja, variasi yang perlu dicek lebih banyak lagi pada kasus ini. Membuat program semacam ini bisa menjadi latihan regex yang bermanfaat. Perintah `/usr/bin/passwd` di Unix biasanya diperlengkapi dengan pengecekan seperti ini. Secara tradisional program pengecek "bad password" ini disebut password cracker atau `crack` untuk singkatnya. Sysadmin dapat menjalankan crack secara berkala pada password semua user, dan memperingati user yang passwordnya dapat dicrack agar menggantinya dengan yang lebih panjang/sulit.

Untuk memperkuat pengecekan password, Anda bisa menambahkan beberapa pemeriksaan tambahan, misalnya dilarang menggunakan nama login atau nama diri sendiri atau nama-nama orang yang sudah umum; dilarang mengandung kata-kata dari kamus lain (kamus merek); dilarang menggunakan tanggal lahir (misalnya: mengandung string 1980 di mana tanggal lahir user adalah 1980); dan lain sebagainya. Tujuan akhirnya adalah, selain melarang pola-pola keteraturan juga melarang penggunaan elemen data-data pribadi yang mudah diketahui orang, seperti tanggal dan tempat lahir, umur, nama binatang peliharaan, alamat rumah, dan lain sebagainya. Tentu saja, Anda terlebih dahulu perlu memiliki data-data user tersebut.

Jangan terlalu "berusaha keras" mencari semua pola yang ada, sebab user Anda akan kesulitan memilih password. Jika pemilihan password terlalu sulit, user akan cenderung mencatat password dan malas mengganti password yang sudah lama tak pernah diganti-ganti.

5-1 Alamat IPv4

Masalah

Anda ingin mengetahui apakah sebuah teks mengandung alamat IP, atau ingin mengecek apakah sebuah string merupakan alamat IP yang valid.

Pola regex

```
# A. ringkas dan sederhana, tapi tidak mengecek alamat seperti
9999.9999.9999.9999
/\d+\.\d+\.\d+\.\d+/
/\d+(\.\d+){3}/

# B. sama seperti A, tapi membatasi tiap bagian maks 3 digit
/\d{1,3}(\.\d{1,3}){3}/

# C. benar-benar mengecek tiap bagian hanya boleh 0-255
/\b(?:25[0-5]|2[0-4][0-9]|1?[0-9][0-9]?)
    (?:\.(?:25[0-5]|2[0-4][0-9]|1?[0-9][0-9]?)){3}\b/x';
```

Diskusi

Sebetulnya untuk mengecek validitas alamat IP dibutuhkan langkah-langkah tambahan. Misalnyaalnya, 0.xx.xx.xx tidak ada. 223.*.*.* ke atas juga masih direserved untuk class D dan E. Sementara kalau Anda ingin mengecek apakah IP ini publik atau tidak, Anda juga harus mencoret 10.xx.xx.xx, 127.xx.xx.xx, dan 192.168.xx.xx.

5-2 Alamat IPv6

Masalah

Anda ingin mengetahui apakah sebuah teks mengandung alamat IP, atau ingin mengecek apakah sebuah string merupakan alamat IP yang valid.

Pola regex

```
A. x:x:x:x:x:x:x:x
/[0-9A-Fa-f]{1,4}(:([0-9A-Fa-f]{1,4})){7}/

B. sama seperti A, tapi dapat mengandung :: (zero-suppressed)
/::|([0-9A-Fa-f]{1,4}||:)(:([0-9A-Fa-f]{1,4}||:)){1,7}/

C. dalam format basis 85
/\b[0-9A-Za-z!#\$%&()*+\-;<=>?\@^_'{|}~]{20}\b/
```

Diskusi

IPv6 adalah penerus bagi IPv4 yang masih banyak digunakan hingga saat ini. Karena range yang meningkat drastis (dari 32-bit untuk IPv4 menjadi 128-bit untuk IPv6) maka untuk merepresentasikan sebuah alamat IPv6 dibutuhkan string yang lebih panjang. Notasi standar IPv6 adalah seperti di pola A, yaitu deretan 8 buah alamat 16-bit dalam heksadesimal. Contohnya:

```
FEDC:BA98:7654:3210:FEDC:BA98:7654:3210
```

Notasi ini cukup panjang, bisa hingga 39 karakter. Salah satu cara mempersingkatnya adalah seperti ditunjukkan di pola B, yaitu menggunakan `::` untuk mewakili deretan bagian yang nilainya 0. Misalnyaalnya, alamat IPv6 berikut:

```
1080:0:0:0:8:800:200C:417A
```

dapat ditulis sebagai:

```
1080::8:800:200C:417A
```

Bahkan

```
0:0:0:0:0:0:0:0
```

dapat ditulis sebagai:

```
::
```

Dan terakhir, IPv6 juga dapat ditulis dalam 20 digit basis 85, seperti ditunjukkan di pola C. Contoh alamat IPv6 dalam notasi ini:

```
4)+k&C#VzJ4br>0wv%Yp
```

Informasi lebih lanjut dapat diperoleh di RFC 1924.

5-3 Hostname/Domain

Masalah

Anda ingin mengecek apakah sebuah string berisi hostname atau nama domain yang valid.

Contoh kode Perl

```
$part_re = qr/[a-z\d]([-a-z\d]*[a-z\d])?/;
$hostname_re = $/^$part_re(\.$part_re)*$/;
```

Diskusi

`tripod.com` memperbolehkan underscore ("_") sebagai subdomain, misalnya: `steven_haryanto.tripod.com`, tapi karakter ini biasanya tidak diterima oleh berbagai software lain di Internet.

Ada kalanya Anda ingin lebih ketat dengan mewajibkan TLD (top level domain) merupakan salah satu yang dikenali. Contoh kodenya ada di http://aspn.activestate.com/ASPN/Cookbook/Rx/Recipe/65123; tidak saya sertakan di sini karena ada dua ratusan TLD yang dikenal.

Untuk Indonesia sendiri, Anda mungkin ingin mengecek SLD (second level domain) yang dikenali:

```
/\.(war\.net|co|net|web|or|ac|sch|mil|go))\.id$/
```

Untuk benar-benar memastikan hostname valid dan hidup (dalam arti, dapat diresolve), Anda dapat mengeceknya dengan perintah host atau bertanya langsung pada DNS cache di jaringan Anda:

```
if ('host '$hostname'' !~ / has address \d+\.\d+\.\d+\.\d+/) {
  print "Maaf, hostname tidak dapat diresolve. Masukkan hostname
lain yang valid";
}
```

Aturan resmi sintaks hostname ada di RFC 1035.

5-4 URL

Masalah

Anda ingin mengecek apakah sebuah string berisi URL yang valid; atau Anda ingin mengekstrak/memroses URL dalam teks.

Pola regex

```
# A. mengerti beberapa scheme umum
/^((?:mailto|https?|ftp|telnet|gopher|news):\S+/

# B. dapat memparse hostname dan path
# $1 berisi scheme (ftp, http, ...)
# $2 berisi username jika ada
# $3 berisi password jika ada
# $4 berisi hostname
# $5 berisi port jika ada
# $6 berisi path
m#^(ftp|https?)://(?:([^:]+)(?::([^@]:)+)?\@)?([^/
]+?)(?::(\d+))?(/\S*)?$#

# C. khusus mengecek sintaks IPv6 dalam URL
# $1 berisi scheme
# $2 berisi IPv6
# $3 berisi nomor port jika ada
# $4 berisi path
$ipv6_re = qr/::|(?:[0-9A-Fa-f]{1,4}||:)(:(?:[0-9A-Fa-
f]{1,4}||:)){1,7}/;
$ipv6_url_re = qr#(ftp|https?)://\[($ipv6_re)\](?::(\d+))?(/
\S*)?$#
```

Contoh kode Perl

```
# membuat URL di dalam teks menjadi hyperlink
s{\b((?:mailto|https?|ftp|telnet|gopher|news):\S+)}{<a
href=$1>$1</a>}g;
```

Diskusi

Sintaks URI dan URL diatur oleh RFC 2396 (untuk sintaks umum URI), RFC 2368 (untuk sintaks mailto:), RFC 2141 (untuk sintaks URN), dan RFC 1808 (untuk sintaks URL relatif). Dilihat dari panjangnya ketiga RFC tersebut, bisa dilihat bahwa aturan sintaks yang ada cukup rumit. URL dimaksudkan sebagai alamat universal yang dapat dipakai untuk segala macam jenis alamat/lokasi, mulai dari nomor telepon `tel:+62211234567` hingga URL Internet. Bahkan Anda dapat menciptakan scheme baru sesuai keinginan sendiri. Contoh salah satu regex lengkap yang mencocokkan URL sesuai aturan RFC ada di http://aspn.activestate.com/ASPN/Cookbook/Rx/Recipe/59864 (regexnya tidak saya sertakan di buku ini karena terlalu panjang).

Untuk sintaks yang lebih sederhana dan umum dijumpai, meskipun tidak 100% sesuai aturan RFC, bisa Anda lihat pada pola-pola di resep ini. Catatan: pola B belum dapat mengecek IPv6 sebab IPv6 mengandung titik dua (bentrok dengan sintaks nomor port) dan dalam URL IPv6 ditulis dengan sintaks:

```
http://[1080::8:800:200C:417A]/
http://[1080::8:800:200C:417A]:80/
```

untuk tujuan mencocokkan IPv6 dalam URL, gunakan pola C. Anda juga bisa menggabungkan pola B dan C menjadi satu tentunya.
Mencocokkan dan memparse URL merupakan salah satu tugas yang amat sering. Bahasa-bahasa seperti Perl, Python, Ruby, dan lain sebagainya. sudah menyediakan modul yang sesuai. PHP sendiri menyediakan fungsi seperti `parse_url()`.

5-5 Escaping URL

Masalah

Anda ingin memasukkan sebuah string ke dalam parameter URL, di mana string tersebut dapat mengandung spasi, dan lain sebagainya.

Contoh kode Perl

```
sub urlencode() {
  my $str = shift;
  $str = s/([^A-Za-z0-9\-_.!~*'()])/sprintf("%%%02X", ord($1))/
eg;
  return $str;
}

# test
for ("atomic group", "atomic!", "atomic???", "&atomic") {
  print "$_ = ", urlencode($_), "\n";
}
```

Output program Perl

```
atomic group = atomic%20group
atomic! = atomic!
atomic??? = atomic%3F%3F%3F
&atomic = %26atomic
```

Diskusi

Proses escaping URL yaitu mengubah karakter-karakter tak aman seperti spasi, ?, dan & (karena ? dan & digunakan sebagai pembatas parameter query string) menjadi %XX di mana XX adalah kode heksadesimal karakter ybs. Dengan di-escape URL, sebuah string menjadi aman untuk dimasukkan ke dalam URL.

Catatan: Perl sudah menyediakan modul `URI::Escape` dan PHP menyediakan `urlencode()`.

5-6 Unescape URL

Masalah

Anda memiliki sebuah parameter URL yang telah diencode. Anda ingin mendecodenya untuk mendapatkan nilai original.

Contoh kode Perl

```
sub urldecode {
  my $str = shift;
  $str =~ s/\+/ /g;
  $str =~ s/%([0-9A-fA-f]{2})/chr(hex($1))/eg;
  return $str;
}

# test
for ("atomic%20group", "atomic!", "atomic%3F%3F%3F", "%26atomic")
{
  print "$_ = ", urldecode($_), "\n";
}
```

Output program Perl

```
atomic%20group = atomic group
atomic! = atomic!
atomic%3F%3F%3F = atomic???
%26atomic = &atomic
```

Diskusi

Proses unescape/decode URL adalah mengubah %XX menjadi karakter dengan kode heksadesimal XX. Sebelumnya kita juga perlu mengubah + menjadi spasi. Karena + merupakan bentuk escaping alternatif bagi spasi. Spasi sering dipakai misalnya dalam query: http://www.google.com/search?q=steven+haryanto. Sehingga selain "%20" disediakan "+" yang lebih singkat.
Catatan: Sebelum zaman Perl versi 5, yakni sebelum modul CGI umum digunakan, program CGI Perl sering melakukan sendiri decoding ini. Ini karena query string yang diperoleh dari webserver di variabel `$ENV{QUERY_STRING}` memang masih terencode.

Catatan: Perl kini sudah menyediakan modul `URI::Escape` dan PHP menyediakan `urldecode()`.

5-7 Escaping HTML

Masalah

Anda ingin mencetak sebuah string ke browser dalam mode HTML, tapi Anda tak ingin tag-tag yang ada pada string berfungsi (dengan kata lain, `"<b>"` pada string dicetak apa adanya sebagai `"<b>"` dan bukan menjadi tag untuk menebalkan teks).

Contoh kode Perl

```
sub htmlencode {
  my $str = shift;
  my %entities = (
    '<' => '&lt;',
    '>' => '&gt;',
    '&' => '&amp;',
    '"' => '&quot;',
  );
  my $entities_re = join "|", map { quotemeta } keys %entities;

  $str =~ s/($entities_re)/$entities{$1}/g;
  return $str;
}

# test
for ("<atomic>", "\"atomic group\"", "&atomic;") {
  print "$_ = ", htmlencode($_), "\n";
}
```

Output program Perl

```
<atomic> = &lt;atomic&gt;
"atomic group" = &quot;atomic group&quot;
&atomic; = &amp;atomic;
```

Diskusi

Proses escaping HTML yaitu mengubah karakter-karakter tak aman di HTML, terutama "<" & ">" (yang dipakai sebagai pembuka & penutup tag) dan "&" (yang dipakai untuk mengawali entiti, seperti `&copy;` atau `&#150;`) menjadi entiti. "<" diubah menjadi "<", "&" menjadi "&", dan seterusnya.
Yang penting diingat di sini adalah pengubahan harus dilakukan secara *paralel*, bukan *seri*. Jadi cara seperti ini salah:


```
$str =~ s/</&lt;/g;   # ganti <
$str =~ s/>/&gt;/g;   # ganti >
$str =~ s/&/&amp;/g;  # ganti &
# dan seterusnya...
```

Bayangkan "<" diganti. Hasilnya yang benar "<". Tapi jika menggunakan kode seri seperti di atas, hasilnya menjadi "<". Pertama "<" diganti menjadi "<" lalu "&" diganti lagi "&". Sehingga string akhir menjadi "<" (salah).
Catatan: Perl sudah menyediakan modul `HTML::Entities` dan PHP menyediakan `htmlentities()`.

5-8 Unescape HTML

Masalah

Anda memiliki string HTML yang terencode, misalnyaalnya "Dharma & Greg". Anda ingin mengubahnya menjadi nilai semula yaitu "Dharma & Greg".

Contoh kode Perl

```
sub htmldecode {
  my $str = shift;
  my %named_ent = (
    'lt' => '<',
    'gt' => '>',
    'amp' => '&',
    'quot' => '"',
    # dan seterusnya...
  );

  $str =~ s[&#(\d+);|&(\w+);]
           [$1 ? chr($1) :
                exists($named_ent{lc $2}) ? $named_ent{lc $2} :
'?']eg;
  return $str;
}

# test
for ("&lt;atomic&gt;", "&quot;atomic group&quot;",
"&amp;atomic;") {
  print "$_ = ", htmldecode($_), "\n";
}
```

Output program Perl

```
&lt;atomic&gt; = <atomic>
&quot;atomic group&quot; = "atomic group"
&amp;atomic; = &atomic;
```

Diskusi

Resep ini padanan dari resep sebelumnya 5-8, yaitu mengubah `&NAMA;` atau `&#NUM;` menjadi huruf entitinya. Namun untuk decode lebih sulit karena ada banyak entiti yang dikenali. Selain `<`, `>`, `&`, `&quot;`, ada banyak lagi misalnya `&copy;` (tanda copyright), ` ` (spasi untuk menyatukan dua buah kata), `&pound;` (tanda poundan seterusnyaerling), dan masih banyak lagi. Daftar lengkap entiti bisa dilihat pada spesifikasi HTML/XHTML pada situs www.w3.org.

Contoh program Perl di resep ini hanya mengenali 4 buah entiti dan jika menjumpai entiti lain akan mengubahnya menjadi `'?'`.

Catatan: Perl sudah menyediakan modul `HTML::Entities` dan PHP menyediakan `html_entity_decode()` yang mengenal daftar entiti dengan lebih lengkap.

5-9 Komentar Dalam HTML

Masalah

Anda ingin mencari atau menghapus komentar dalam HTML.

Pola regex

```
/<!–(.*?)–>/s
```

Penjelasan

Komentar dalam HTML dan XML ditulis dalam tanda `<!–` dan `–>`. Non-greedy `.*?` di sini penting sebab tanpa non-greedy, Anda bisa salah melahap kode HTML yang berada di antara komentar, misalnya:

```
<!– Contoh komentar. –><html><head><title>Title...</title></head>
<body><!– komentar 2 –>
```

5-10 Strip HTML

Masalah

Anda ingin membuang semua tag-tag HTML pada teks.

Contoh kode Perl

```
# tahap 1. buang semua komentar
s/<!–(.*?)–>//sg;

# tahap 2. buang semua tag lain
s/<[^>]*>//sg;
```

Diskusi

Menstrip HTML biasanya diperlukan jika kita ingin menyaring masukan dari form HTML, atau saat ingin mencari teks dalam HTML. Karena diselingi tag, kita tidak dapat mencari teks "kerupuk" dalam dokumen HTML seperti di bawah:

```
<font size=3>K</font>erupuk ikan.
```

Setelah di-strip, dokumen menjadi siap dicari:

```
Kerupuk ikan.
```

Mengapa kita harus melakukan dalam dua tahap, kenapa tidak bisa dengan sekali operasi `s/<[>]+>//sg;` saja? Sebab komentar dalam HTML dapat mengandung tag-tag lain, jadi buang dulu komentar, baru tag-tag lain di luar komentar.

Catatan: Regex di resep ini tidak dapat untuk semua kasus, karena dalam sebuah tag, dapat saja ditemukan karakter `">"` yang bukan penutup tag. Misalnya:

```
<img src="steven.jpg" alt="<<<Cakep gak??>>>">
```

Dengan pola regex `/(<[^>]*>)/`, yang tertangkap adalah:

```
<img src="steven.jpg" alt="<<<Cakep gak??>
```

Pada kasus ini, Anda harus memparse tag dan atribut, misalnya dengan parser HTML.

Catatan: PHP sudah menyediakan fungsi `strip_tags()`.

5-11 Memangkas HTML

Masalah

Anda memiliki dokumen HTML yang terlalu besar atau *bloated* atau *evil*, Anda ingin memperkecil/memangkasnya.

Contoh kode Perl

```
# A. membuang Javascript
s#<script\b.+?</script>##sig;

# B. membuang tag CSS
s#<link\b.+?>##sig;

# C. membuang font, termasuk tag penutupnya
s#</?font\b.*?>##sig;

# D. membuang beberapa tag sekaligus, img, font, dan blink
s#</?(font|img|blink)\b.*?>##sig;

# E. membuang semua sebelum <body> dan sesudahnya
s#^.+(<body\b)#$1#si;
s#(</body>).+#$1#si;

# F. membuang semua sebelum <body> dan sesudahnya, termasuk
<body> itu sendiri
s#^.+(<body\b)##si;
s#(</body>).+##si;
```

Diskusi

Catatan: PHP menyediakan fungsi `strip_tags()`.

6-1 Alamat E-mail

Masalah

Anda ingin mengecek apakah sebuah string berisi alamat e-mail yang valid.

Pola regex

```
# A. varian 1
/^[\w.-]+\@\w+(\.\w+)*$/

# B. varian 2
$user = '[\x21\x23-\x27\x2A-\x2B\x2D\w\x3D\x3F]+'.
        '(?:\.[\x21\x23-\x27\x2A-+\x2B\x2D\w\x3D\x3F]+)*';
$ip = '\[\d{1,3}(?:\.\d{1,3}){3}\]';
```



```
$hostname = '[A-Za-z0+-9][A-Za-z0-9-]*'.
            '(?:\.[A-Za-z0-9][A-Za-z0-9-]*)*';
$regex = /^$user\@(?:$ip|$hostname)$/;
```

Diskusi

Sintaks alamat email diatur oleh RFC 822 (yang telah diperbarui oleh RFC 2822). Aturan lengkap RFC juga cukup rumit, menyebabkan regex alamat email yang 100% mengikuti RFC panjang sekali (bisa Anda lihat di buku *Mastering Regular Expressions* atau di source code salah satu modul Perl seperti `RFC::RFC822::Address` misalnya). Pola yang diberikan di resep ini sendiri sudah cukup untuk mayoritas alamat email yang ditemui.

Memeriksa sintaks alamat email juga salah satu pekerjaan yang amat sering harus dilakukan, karenanya sebetulnya Anda tidak perlu beregex ria lagi sebab rata-rata bahasa sudah menyediakan modul/fungsinya. Perl misalnyaalnya menyediakan banyak modul: `RFC::RFC822::Address`, `Email::Valid`, `Mail::CheckUser`; PHP menyediakan fungsi `imap_rfc822_parse_adrlist()`.

6-2 Jumlah Email di Mailbox

Masalah

Anda ingin dengan cepat mengetahui berapa email yang ada di sebuah mailbox.

Contoh perintah grep

```
$ grep -c '^From ' FILEMBOX
```

Diskusi

Mailbox dengan format mbox yaitu setiap email diawali dengan baris seperti ini:

```
From - Thu Jan 08 09:37:14 2004
```

jadi untuk menghitung email mudah saja, yaitu mencari baris pola `/^From /`. Bagaimana jika ada e-mail yang dimulai "From". Secara teori tidak boleh, jadi biasanya program email harus mengubah jadi " From " (diawali spasi), atau ">From " (diawali ">") dan sebagainya. Format mbox dipakai antara lain oleh Eudora, Mozilla Mail, Evolution, dan Thunderbird. Ada lagi format popular bernama Maildir, di mana setiap email ditaruh dalam file terpisah. Untuk menghitung jumlah e-mail dalam mailbox juga mudah, tinggal menghitung jumlah file yang ada:

```
$ ls /path/ke/Maildir/new | wc -l
```

6-3 E-mail Harvester

Masalah

Anda ingin mengekstrak alamat e-mail dari HTML, teks, mailbox, dan sebagainya.

Contoh kode Perl

```
  1|#!/usr/bin/perl
   |
  3|#
  4|# email-harvester.pl
  5|#
   |
  7|%emails = ();
   |
  9|$user_re     = '[\x21\x23-\x27\x2A-\x2B\x2D\w\x3D\x3F]+'.
 10|               '(?:\.[\x21\x23-\x27\x2A-
   +\x2B\x2D\w\x3D\x3F]+)*';
 11|$ip_re       = '\[\d{1,3}(?:\.\d{1,3}){3}\]';
 12|$hostname_re = '[A-Za-z0+-9][A-Za-z0-9-]*'.
 13|               '(?:\.[A-Za-z0-9][A-Za-z0-9-]*)*';
 14|$email_re    = qr/\b$user_re\@(?:$ip_re|$hostname_re)\b/o;
   |
 16|while (<>) {
 17|  while (/($email_re)/g) {
 18|    $emails{lc $1}++;
 19|  }
 20|}
   |
 22|for (sort keys %emails) {
 23|  print "$_\n";
 24|}
   |
 26|print "=" x 72, "\n";
 27|print "Total: ", scalar(keys %emails), " alamat email\n";
```

Contoh output program

```
$ ./email-harvester.pl /home/steven/mbox
...
... (daftar alamat email dicetak...)
...
========================================================================
Total: 386 alamat email
```

Diskusi

Resep ini merupakan aplikasi resep 6-1, yaitu mencocokkan alamat email dan mengambil match groupnya. Alamat email disimpan dalam hash dalam bentuk huruf kecil semua untuk mencegah duplikasi.

6-4 E-mail Obfuscator

Masalah

Anda tidak ingin alamat email yang Anda cantumkan diharvest orang, jadi Anda ingin memodifikasi tampilan atau menyembunyikan alamat email misalnyaalnya "steven@domain.com" menjadi "steven at domain dot com" atau "steven@d...", dan lain sebagainya.

Contoh kode Perl

```
$user_re     = '[\x21\x23-\x27\x2A-\x2B\x2D\w\x3D\x3F]+'.
               '(?:\.[\x21\x23-\x27\x2A-+\x2B\x2D\w\x3D\x3F]+)*';
$hostname_re = '[A-Za-z0+-9][A-Za-z0-9-]*'.
               '(?:\.[A-Za-z0-9][A-Za-z0-9-]*)*';
$email_re    = qr/\b$user_re\@$hostname_re\b/o;

# A. mengubah "steven@domain.com" menjadi "steven@HIDDEN"
s/\b($user_re)\@$hostname_re\b/$1\@HIDDEN/g;

# B. mengubah "steven@domain.com" menjadi "steven@d..."
s[\b($user_re)\@($hostname_re)\b]
 ["$1\@" . substr($2,0,1) . "..."]eg;

# C. mengubah "steven@domain.com" menjadi "steven at domain dot
com"
sub ganti_dot { my $str = shift; $str =~ s/\./ dot /g; return
$str }
s/\b($user_re)\@$hostname_re\b/"$1 at ".ganti_dot($2)/eg;

# D. mengubah "steven@domain.com" menjadi
"steven@NOSPAM.domain.com"
s/\b($user_re)\@$hostname_re\b/$1\@NOSPAM.$2/g;
```

```
# E. mengubah "steven@domain.com" menjadi "&#115;&#116;&#101;..."
#    alamat email akan tampil di browser seperti biasa, namun
sebagian
#    email harvester akan mengalami kesulitan melihatnya
sub htmlescapeall { my $s = shift; $s =~ s/(.)/"&#".ord($1).";"/
eg; $s; }
s/\b($user_re\@$hostname_re)\b/htmlescapeall($1)/eg;
```

7-1 Headline Berita Detikcom

Masalah

Anda ingin mengambil headline berita dari situs http://www.detik.com.

Contoh kode PHP

```
 1|<?
  |
 3|//
 4|// headline-detik.php
 5|//
  |
 7|if (($fp = fopen("http://www.detik.com/
  index.htm","r"))===false)
 8|  die("Gagal mengambil URL!");
  |
10|$content = "";
11|while ($chunk = fgets($fp)) $content .= $chunk;
  |
13|if (!$content)
14|  die("Dokumen URL kosong!");
  |
16|if (!preg_match_all('#<span
  class="tanggal(?:High)?">([^>]+)</span><BR>'  .
17|      '\s*<A href="([^">]+)"[^>]+><span
  class="(?:judulHigh|strJudul)?">'  .
18|      '(.+?)</span></A><BR>\s*<span
  class="summary(?:High)?">'  .
19|      '(.+?)</span>#si', $content, $mm, PREG_SET_ORDER))
20|  die("Gagal mengekstrak headline dari dokumen URL!");
```



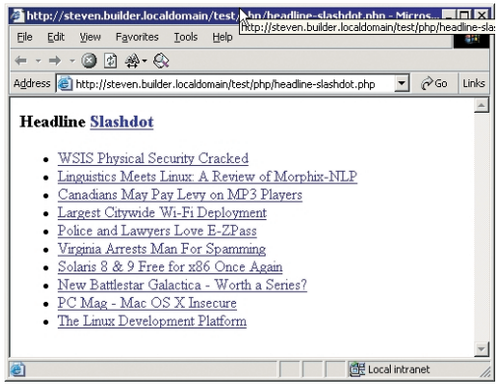
```
  |
22|$items = array();
23|foreach ($mm as $m) {
24|  $items[] = array(
25|    'date' => $m[1],
26|    'link' => $m[2],
27|    'title' => $m[3],
28|    'summary' => $m[4],
29|  );
30|}
  |
32|echo "<h3>Headline <a href=\"http://www.detik.com\">".
33|     "detikcom</a></h3>";
34|echo "<ul>\n";
  |
36|foreach ($items as $item) {
37|  echo "<li><a href=$item[link]>$item[title]</a></li>";
38|}
  |
40|echo "</ul>";
  |
42|?>
```

Contoh output program

Diskusi

Setelah mempelajari regex, Anda akan melihat bahwa untuk mengambil headline berita, kurs, harga saham, dan lain sebagainya dari Internet begitu mudah. Semua informasi ini tersedia bebas dan kita tinggal mengekstrak informasi yang ada dari halaman HTML. Aktivitas seperti ini biasanya disebut grabbing atau HTML scraping ("mengerik sesuatu yang berharga dari halaman HTML"). Hanya saja, yang perlu diperhatikan: 1) term of use, beberapa situs, termasuk detikcom, nampaknya keberatan jika kita melakukan grabbing lalu menampilkan hasilnya di situs lain. Sebab menurut mereka ini dapat mengurangi pageview (dan ujung-ujungnya mengurangi mereka yang melihat banner di halaman detikcom); 2) dari waktu ke waktu situs diredesain sehingga kode HMTL berubah dan pola regex kita *broken*, dan kita harus merumuskan pola regex baru.

Saya menyertakan tiga resep di 7-1, 7-2, 7-3. Karena prinsip kodenya sama semua (ambil halaman HTML, masukkan ke variabel, cocokkan dengan regex, ambil match group). Saya hanya menampilkan potongan halaman HTML (dengan data yang ingin diambil dalam cetak tebal dan bergaris bawah) serta pola regexnya saja.



Gambar 4-10.

7-2 Headline Berita DetikInet

URL

http://www.detikinet.com/index.shtml

Potongan HTML

```
...
<table width="358" border="0" cellspacing="0" cellpadding="4"
align="center">
<tr><td align="left" valign="top" id="tblHiglight">
<span class="tanggalHigh">Jumat, 12/12/2003 11:09 WIB</span><BR>
<A href="http://www.detikinet.com/net/2003/12/12/20031212-
110911.shtml"
class="hlhigh" target="_self"><span class="judulHigh">Proyek
Sosial TI Raih 'Cyber Oscar'</span></A><BR>
<span class="summaryHigh">Sebuah proyek sosial TI, meraih
penghargaan Cyber Oscar. Proyek ini mewujudkan kepeduliannya
terhadap para korban ranjau darat di Kamboja dengan menyediakan
lapangan pekerjaan bidang teknologi informasi (TI).</span><HR
class="myhr">
</td></tr></table><br>
...
```

Pola regex

```
m#<span\sclass="tanggal(?:High)?">([^>]+)</span>
  <BR>
  \s*<A\shref="([^"]+)"[^>]+>
  <span\sclass="(?:judulHigh|strJudul)?">(.+?)</span>
  </A>
  <BR>
  \s*<span\sclass="summary(?:High)?">
  (.+?)</span>
 #sixg
```


7-3 Headline Berita Slashdot

URL

http://slashdot.org/slashdot.rdf

Potongan HTML

```
<?xml version="1.0" encoding="ISO-8859-1"?>

<rdf:RDF
xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
xmlns="http://my.netscape.com/rdf/simple/0.9/">

<channel>
<title>Slashdot</title>
<link>http://slashdot.org/</link>
<description>News for nerds, stuff that matters</description>
</channel>

<image>
<title>Slashdot</title>
<url>http://images.slashdot.org/topics/topicslashdot.gif</url>
<link>http://slashdot.org/</link>
</image>

<item>
<title>Scientists Freeze Pulse Of Light</title>
<link>http://slashdot.org/article.pl?sid=03/12/11/0023218</link>
</item>

<item>
<title>Australian Pilot Stranded In Antarctica</title>
<link>http://slashdot.org/article.pl?sid=03/12/10/2050256</link>
```

```
</item>

...
```

Pola regex

```
m#<item>\s*<title>(.+?)</title>\s*<link>(.+?)</link>\s*</
item>#sig
```

7-4 Kurs Dolar Yahoo!

URL (mengubah dari dolar ke rupiah)

http://finance.yahoo.com/m5?a=1&s=USD&t=IDR

Potongan HTML

```
Symbol</th><th align=center>U.S. Dollar</th><th colspan=2
align=center>Exchange Rate</th><th align=center>Indonesian
Rupiah</th><th align=center>Bid</th><th align=center>Ask</th><th
align=center>Historical Charts</th></tr><tr align=center><th><a
href="/q?s=USDIDR=X&d=c">USDIDR=X</a></th><th>1</th><th
nowrap>Jan  9</th><td>8320.0000</td><td><b>8,320</b></
td><td>8320.0000</td><td>8340.0000</td><td>3m, <a href="/
m5?s=USD&t=IDR&a=1&c=1">1y</a>, <a href="/
m5?s=USD&t=IDR&a=1&c=2">2y</a>, <a href="/
m5?s=USD&t=IDR&a=1&c=3">max</a></td></tr><tr align=center
valign=top><td colspan=8 bgcolor=white><img border=0 src="http://
ichart.yahoo.com/z?
```

Pola regex

```
m#<th  nowrap>(\d+:\d+[ap]m)</th><td>(\d+\.\d+)</td>#
```

8-1 Komentar dalam Bahasa Pemrograman

Masalah

Anda ingin mengenali/mengekstrak/memroses komentar dalam berbagai bahasa pemrograman.

Pola regex

```
# A. Gaya shell atau Perl/PHP/Python/Ruby (# ... hingga akhir
baris)
/\#.*/

# B. Gaya C (diapit tanda /* ... */)
/\*.*?\*/s

# B2. sama seperti B, tapi mendukung nested comment.
# ditulis dalam sintaks Perl
$c_comment = qr{/\*((?>[^*/]+)|(?p{$c_comment}))*\*/}

# B3. sama seperti B2, tapi ditulis dalam sintaks PCRE.
{/\*((?>[^*/]+)|(?1))*\*/}

# C. C++/Java/Delphi (//... hingga akhir baris)
m{//.*}

# D. Turbo Pascal (diapit { ... })
/\{.+?\}/s
/\{[^}]*\}/s

# E. Pascal tradisional (diapit (* ... *))
/\(\*.*?\*\)/s

# F. VB/ASP/batch file DOS (' ... hingga akhir baris)
/'.*/
```

Contoh kode Perl 1 (menghitung jumlah baris kode)

```
  1|#!/usr/bin/perl
   |
  3|#
  4|# perl-loc.pl
  5|#
   |
  7|%files = ();
  8|$current_file = "";
   |
 10|$files = 0;
```

```
 11|$loc = 0;
 12|$lines = 0;
 13|$current_file = "";
   |
 15|while (<>) {
 16|  $lines++;
 17|  if ($ARGV ne $current_file) { $files++; $current_file =
   $ARGV }
   |
 19|  next unless /\S/;
 20|  next if /^\s*#/;
 21|  $loc++;
 22|}
   |
 24|print <<END;
 25|Files: $files
 26|Lines: $lines
 27|kLOC : ${\( sprintf("%.2f", $loc/1000) )}
 28|END
```

Contoh output kode Perl 1

```
$ ./perl-loc.pl *.pl
Files: 20
Lines: 854
kLOC : 0.63
```

Contoh kode Perl 2 (syntax highlighter PHP sederhana)

```
  1|#!/usr/bin/perl
   |
  3|#
  4|# simple-php-highlight.cgi
  5|#
   |
  7|use CGI ':standard';
  8|use HTML::Entities;
   |
 10|sub cek_token {
 11|  my ($token, $state) = @_;
   |
```



```
 13|  if ($state->{s1} eq 'html') {
 14|    if ($token =~ /<\?php|<\?=|<\?/) {
 15|      $state->{s1} = 'php';
 16|    }
 17|  } else {
 18|    if ($token eq '?>' && $state->{s2} !~ /^(com|quote)\./) {
 19|      $state->{s1} eq 'html';
 20|    } elsif ($token eq '//' && $state->{s2} !~ /
   ^(com|quote)\./) {
 21|      $state->{s2} = "com.cpp";
 22|    } elsif ($token eq '#' && $state->{s2} !~ /
   ^(com|quote)\./) {
 23|      $state->{s2} = "com.sh";
 24|    } elsif ($token eq '/*' && $state->{s2} !~ /
   ^(com|quote)\./) {
 25|      $state->{s2} = "com.c";
 26|    } elsif ($token eq '*/' && $state->{s2} !~ /
   ^(com\.(sh|cpp)|quote\.)/) {
 27|      $state->{s2} = "";
 28|    } elsif ($token eq '"' && $state->{s2} !~ /
   ^(com\.|quote\.s)/) {
 29|      $state->{s2} = $state->{s2} eq 'quote.d' ? '' :
   'quote.d';
 30|    } elsif ($token eq "'" && $state->{s2} !~ /
   ^(com\.|quote\.d)/) {
 31|      $state->{s2} = $state->{s2} eq 'quote.s' ? '' :
   'quote.s';
 32|    } elsif ($token =~ /[\015\012]/ && $token !~ /
   ^(quote\.|com\.c\z)/) {
 33|      $state->{s2} = "" if $state->{s2} =~ /^com\.(sh|cpp)/;
 34|    }
 35|  }
   |
 37|  "<span class=".(
 38|    ($state->{s1} eq 'html' ? "html" :
 39|      ($state->{s2} =~ /^com\./ ? 'com' :
 40|        ($state->{s2} =~ /^quote\./ ? 'str' : 'php')
 41|      )
 42|    )
 43|  .">".
 44|  #"[$state->{s1},$state->{s2},". # DEBUGGING
```

```
 45|  encode_entities($token).
 46|  #"]". # DEBUGGING
 47|  "</span>";
 48|}
   |
 50|print "Content-Type: text/html\n\n";
   |
 52|if (param('teks')) {
 53|  $teks = param('teks');
   |
 55|  %state = (
 56|    s1 => "html", # "html" or "php"
 57|    s2 => "",     # "", "quote.s", "quote.d", "com.c",
   "com.cpp", "com.sh"
 58|  );
   |
 60|  $teks =~
 61|  s{(
 62|     <\?php|<\?=|<\?|        # tag pembuka php
 63|     \?>|                    # tag penutup php
 64|     //|                     # tanda pembuka komen C++-style
 65|     \043|                   # tanda pembuka komen shell-
   style
 66|     /\*|                    # tanda pembuka komen C-style
 67|     \*/|                    # tanda penutup komen C-style
 68|     \047|                   # kutip tunggal
 69|     \042|                   # kutip ganda
 70|     [\015\012]+|            # newline
 71|     \\.|                    # backslash
 72|     [^</\043*\042\047\015\012\\]+|
 73|     <+|\*+|/+               # bukan salah satu token di atas
 74|    )
 75|   }
 76|   {cek_token($1, \%state)}egx;
   |
 78|   print "
 79|     <style>
 80|       .php  {color: #000000}
 81|       .html {color: #666666; background-color: #cccccc}
 82|       .str  {color: #3333ff; font-weight: bold}
 83|       .com  {color: #669966; font-style: italic}
```



```
 84|     </style>
 85|  ";
   |
 87|   print "<pre>", $teks, "</pre>";
   |
 89|} else {
   |
 91|  print "
 92|    <form method=POST>
 93|    <textarea name=teks cols=80 rows=6></textarea><br>
 94|    <input type=submit>
 95|    </form>
 96|  ";
   |
 98|}
```

Pembahasan

Tiap bahasa pemrograman punya sintaks dan aturan masing-masing untuk komentar. Salah satu perbedaan yang paling penting adalah apakah sebuah komentar dapat *nested* (bersarang). Nested maksudnya komentar di dalam komentar. Contoh, dengan gaya C:

```
/* ini sebuah komentar. di dalam komentar ini ada
   komentar lain: /* ... komentar level 2 ... */
   ini komentar level 1 lagi.
*/
```

Standar C ANSI tidak memperbolehkan komentar nested, tapi beberapa varian C dan Pascal lain memperbolehkan ini.

Dengan regex Perl dan PCRE kita dapat mencocokkan pola-pola nested seperti ini dengan *pola rekursif*. Lihat pola B2 dan B3. Di situ Anda akan melihat subpola:

```
(?p{$c_comment})
```

dan

```
(?1)
```

di mana `$c_comment` atau `(?1)` akan digantikan dengan keseluruhan pola B2 dan B3 itu sendiri. Dengan menggunakan pola rekursif, maka komentar bersarang akan

dapat diambil dengan benar, sementara jika menggunakan pola B terhadap string komentar di atas, yang tertangkap adalah:

```
/* ini sebuah komentar. di dalam komentar ini ada
   komentar lain: /* ... komentar level 2 ... */
```

Dengan kata lain, begitu menemukan */ pertama, pencocokan selesai.

perl-loc.pl. Mari membahas contoh program pertama. Program sederhana ini menghitung jumlah baris code (LOC, line of code), yaitu jumlah baris file yang tidak kosong dan bukan sepenuhnya komentar. Untuk bahasa-bahasa tertentu yang menggunakan komentar 1 baris (ala # shell atau // C++) maka untuk melakukan perhitungan LOC cukup mudah, bisa dilakukan per baris. Tapi jika bahasa pemrogramannya memperbolehkan komentar multibaris, apalagi komentar nested, maka Anda perlu melakukan parsing atau bisa juga memproses filenya dulu dengan menstrip semua komentar yang ada sebelum menghitung LOC.

Program **perl-loc.pl** tentu saja amat simplistik. Program ini tidak memperhitungkan kasus di mana ada baris yang merupakan bagian dari string, misalnya:

```
print "
  # hi, ini bukan komentar tapi bagian dari string
";
```

Namun biasanya memang untuk menghitung ini LOC boleh-boleh saja. Yang penting adalah selama proyek berjalan algoritma/cara menghitung LOC tetap sama. LOC dipakai untuk menjadi gambaran kira-kira ukuran kode dan perkembangan proyek.

simple-php-highlight.cgi. Mari bahas program kedua. Program CGI Perl ini dapat dipakai untuk melakukan highlighting sederhana pada sebuah kode sumber PHP, yaitu memberi warna abu-abu pada HTML (bukan kode PHP), warna biru pada string, dan warna hijau pada komentar. Outputnya bisa dilihat seperti pada Gambar 4-11.

Program kedua ini adalah contoh lain melakukan parsing dengan regex, mirip dengan resep 3-2. Pertama kita menyiapkan state (%state) lalu mengecek setiap token. Token yang kita pilih antara lain: kutip, tag pembuka/penutup PHP, newline, serta karakter-karakter pembuka dan penutup komentar. Setiap ditemukan token, state potensial berubah. Misalnyaalnya, kita masuk ke dalam mode PHP manakala menjumpai tag pembuka PHP. Atau kita masuk ke dalam komentar jika menjumpai

token "//", dan mengakhiri komentar 1 baris jika menjumpai newline. Tapi dalam mengubah state, kita juga harus melihat state (konteks) yang ada saat itu. Misalnyaalnya, jika kita menemukan token tag penutup PHP ?> maka belum tentu kita keluar dari mode PHP; kalau kita ada di dalam string, maka token tersebut merupakan bagian dari string. Demikian juga jika kita menjumpai kutip di dalam komentar, kita tidak menganggap kutip tersebut membuka state string.

Tentu saja program highlighter ini masih simplistik dan jauh dari lengkap/sempurna.

8-2 Nama Variabel

Masalah

Anda ingin mengetahui apakah sebuah string merupakan nama variabel yang sah.

Pola regex

```
/^[A-Za-z_][A-Za-z0-9_]*/
```

Diskusi

Di rata-rata bahasa pemrograman, nama variabel haruslah dimulai dengan huruf atau underscore ("_") dan boleh diikuti huruf, underscore, dan angka. Bahasa-bahasa tertentu membatasi panjang variabel. Misalnyaalnya, jika panjang maksimum 31 karakter:

```
/^[A-Za-z_][A-Za-z0-9_]{0,30}$/
```

Jika string lebih panjang dari 31 karakter, maka pola di atas tidak akan cocok. Bahasa tertentu bisa juga mewajibkan variabel dimulai dengan huruf kecil, sebab prefiks huruf besar dipakai untuk konstanta.

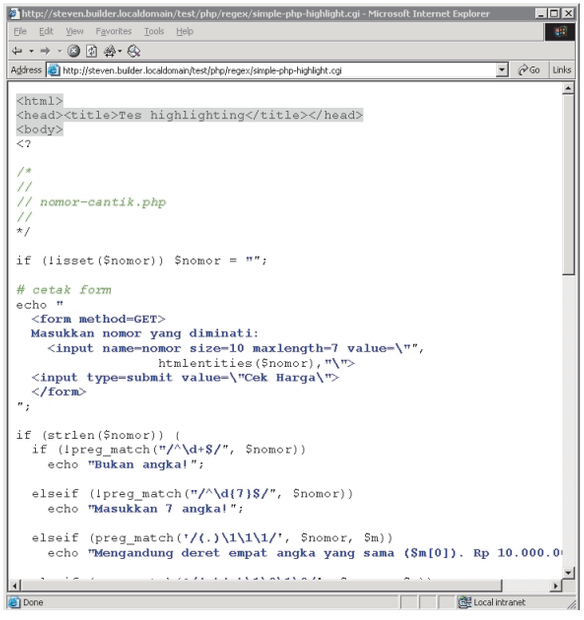
```
/^[a-z_][A-Za-z0-9_]*$/
```

Bahasa lain seperti PHP dan Perl memiliki prefiks $:

```
/^\$[A-Za-z_][A-Za-z0-9_]*$/
```

Bahasa Perl memperbolehkan variabel ditulis dalam format lengkap (fully-qualified) dengan nama package-nya, misalnya: `$main::foo` atau `$HTML::Entities::bar`:

```
/^\$([A-Za-z_][A-Za-z0-9_]*::)*[A-Za-z_][A-Za-z0-9_]*/
```



Gambar 4-11.

dan tentu saja, bahasa seperti Perl, shell, dan Ruby memiliki variabel-variabel khusus seperti $0, $1, $&, $/, dan lain sebagainya. Biasanya kita jarang melakukan pengecekan ini karena kasus terbanyak adalah kita ingin mengecek validitas simbol buatan user dan bukan variabel builtin tertentu dari bahasa pemrograman.

8-3 Substitusi Template

Masalah

Anda ingin membuat sistem template sendiri.

Contoh kode PHP

```
 1|<?
  |
 3|//
 4|// template.php
 5|//
  |
 7|if (!isset($submit)) $submit = "";
 8|if (!isset($teks)) $teks = "";
```



```
 9|if (!isset($keys)) $keys = array("var1", "var2", "var3",
  "var4", "var5");
10|if (!isset($values)) $values = array("1", "2", "3", "4",
  "5");
  |
12|echo "
13|  <form method=POST>
14|  Template:<br>
15|  <textarea name=teks cols=70
  rows=7>",htmlentities($teks),"</textarea>
16|  <br>
17|";
  |
19|  echo "Variabel template:<br>";
20|  for ($i=0; $i<count($keys); $i++) {
21|    echo "
22|      <input name=keys[]
  value=\"",htmlentities($keys[$i]),"\">
23|      <input name=values[]
  value=\"",htmlentities($values[$i]),"\">
24|      <br>
25|    ";
26|  }
  |
28|echo "
29|  <p>Kelakuan jika sebuah variabel template tidak
  ditemukan:<br>
30|    <input type=radio name=on_undefined value=error>Pesan
  error
31|    <input type=radio name=on_undefined value=ignore
  checked>Biarkan
32|    <input type=radio name=on_undefined value=empty>Kosongkan
33|  <p>
34|  <input type=submit name=submit>
35|  </form>
36|";
  |
38|if ($submit) {
39|  $vars = array();
40|  for ($i=0; $i<count($keys); $i++) {
41|    $vars[ $keys[$i] ] = $values[$i];
```

```
 42|  }
   |
 44|  echo "<h3>Hasil substitusi</h3>";
   |
 46|  if ($on_undefined == "error")
 47|    $r = "isset(\$vars['\\1']) ? \$vars['\\1']:'[ERROR:undef
   '.'\\1'.']'";
 48|  elseif ($on_undefined == "ignore")
 49|    $r = "isset(\$vars['\\1']) ?
   \$vars['\\1']:'['.'\\1'.']'";
 50|  else // empty
 51|    $r = "isset(\$vars['\\1']) ? \$vars['\\1']:''";
   |
 53|  echo "<pre>";
 54|  echo htmlentities(
 55|    preg_replace("/\[(\w+)\]/e", $r, $teks)
 56|  );
 57|  echo "</pre>";
 58|}
   |
 60|?>
```

Contoh output program

Diskusi

Template sering digunakan dalam pemrograman web, karena dapat membantu memisalnyaahkan logic dan presentation. Untuk membuat sistem template sederhana sendiri, caranya mudah saja dengan regex. Kode PHP pada contoh akan mengganti semua bentuk:

```
[VARNAME]
```

pada teks dengan isi dari variabel `$vars['VARNAME']`. Anda tentu bisa memilih sintaks sendiri, misalnya:

```
$VARNAME
```

dengan regex

```
/\$(\w+)/
```

atau


```
{{VARNAME}}
```

dengan regex

```
/\{\{(\w+)\}\}/
```

dan lain sebagainya.

Ada beberapa isu yang perlu diperhatikan. **Pertama**, jika template mengandung sebuah variabel yang tidak terdefinisi di dalam array `$vars`, apa yang harus/ingin dilakukan? Program contoh memberi tiga pilihan, dan masing-masing pilihan bisa Anda lihat kodenya di baris 46-51.

Kedua, apakah pencocokan nama variabel ingin case-sensitive atau tidak? Pada program contoh case-sensitive. Jika Anda ingin tidak case-sensitive, maka nama variabel di $vars disimpan dalam huruf kecil semua, lalu Anda selalu menggunakan

Gambar 4-12.

`\$vars[strtolower('\\1')]` di baris 46-51.

Bagaimana jika kita ingin membuat sistem template yang mengandung perintah seperti kondisional (IF), looping (FOR), atau include? Ada beberapa cara. Salah satunya adalah: 1) pecah-pecah dulu template menjadi token; setiap token adalah literal atau perintah template; 2) parse setiap perintah template, misalnya:

```
if (preg_match("/^include (\S+)$/", $command, $m))
  // cek file $m[1] dan load jika ada
```

```
elseif (preg_match("/^if (\S)$/", $command, $m))
  // proses perintah IF
...
elseif (preg_match("/^\$(\w+)$/", $command, $m))
  // substitusikan variabel $vars[ $m[1] ]
```

8-4 GUID Hexstring

Masalah

Anda ingin mengetahui apakah sebuah string merupakan GUID hexstring dengan format yang benar.

Pola regex

```
/^[0-9a-f]{8}-[0-9a-f]{4}-[0-9a-f]{4}-[0-9a-f]{4}-[0-9a-f]{12}$/i
```

Diskusi

GUID adalah data 128-bit yang biasanya dinyatakan dalam notasi heksadesimal sebagai berikut `a6c1a7f8-a7f3-4bf2-953a-6cd3 60e2 618d`.

8-5 Tag CVS/Subversion

Masalah

Anda ingin mengekstrak informasi tag CVS/Subversion yang terdapat dalam file.

Pola regex

```
/\$(\w+): (.+?)\$/
```

Contoh kode Perl

```
  1|#!/usr/bin/perl
   |
  3|#
  4|# extract-cvs-tag.pl
  5|#
   |
  7|use Tie::InsertOrderHash;
   |
  9|$current_file = "";
 10|tie %files, 'Tie::InsertOrderHash';
```



```
   |
 12|while (<>) {
 13|  # telah berganti file
 14|  if ($ARGV ne $current_file) {
 15|    $current_file = $ARGV;
 16|    $files{$current_file} = {};
 17|  }
 18|  while (/\$(\w+): (.+?)\$/g) {
 19|    $files{$current_file}{$1} = $2;
 20|  }
 21|}
   |
 23|# report
 24|for $file (keys %files) {
 25|  print "$file:\n";
 26|  while (($k, $v) = each %{ $files{$file} }) {
 27|    print "  $k = $v\n";
 28|  }
 29|}
```

Contoh output program

```
$ ./extract-cvs-tag.pl /tmp/*.cgi
/tmp/click.cgi:
  Id = click.cgi 45 2003-12-31 07:25:48Z adserver
/tmp/pimage.cgi:
  Id = pimage.cgi 6 2003-10-24 10:44:55Z adserver
/tmp/serve.cgi:
  Id = serve.cgi 43 2003-12-24 23:21:30Z adserver
```

Diskusi

Yang dimaksudkan dengan tag CVS adalah keyword dalam format seperti berikut ini:

```
$Id$
$Author$
$Version$
```

String ini dapat ditanamkan di dalam komentar program:

```
/* $Id$ */
```

atau di dalam string:

```
($version) = '$Version$' =~ /(\d+(\.\d+)+)/;
```

String seperti ini oleh CVS/Subversion akan disubstitusi dengan nilai sebenarnya setelah kita melakukan commit. Misalnyaalnya menjadi:

```
$Id: click.cgi 45 2003-12-31 07:25:48Z adserver $
```

Cara ini umum digunakan untuk memberi tanda/angka versi pada file; nilai akan diupdate otomatis oleh tool revision control tanpa harus kita edit manual.

8-6 Membuat Password Random Yang Manusiawi

Masalah

Anda ingin menggenerate password acak, tapi tidak ingin yang benar-benar acak seperti "xoiunmsauyv" atau "1023998" karena sulit diingat manusia; melainkan password yang mengandung kata-kata kamus seperti "robotic" atau "saddle9734".

Contoh kode PHP

```
 1|<?
  |
 3|//
 4|// random-password.php
 5|//
  |
 7|$Words = array();
  |
 9|// ambil kata dari kamus
10|if (($fp = @popen("grep '^.......\\?.\\?$' /usr/share/dict/
  words",
11|                 "r"))!== false) {
12|  while ($line = fgets($fp, 8192)) {
13|    $Words[ strtolower(trim($line)) ] = 1;
14|  }
15|}
  |
17|function random_password() {
18|  global $Words;
19|  return array_rand($Words) . rand(1000,9999);
20|}
```



```
  |
22|// test
23|echo random_password();
  |
25|?>
```

Diskusi

Program contoh di atas hanya berjalan di Unix/Linux, karena distro-distro Linux umumnya menyediakan kamus di lokasi `/usr/share/dict/words`, berupa file berisi kata-kata bahasa Inggris, satu kata per baris.

Inti program adalah baris 10, di mana kita menggunakan grep untuk memilih kata 6 hingga 8 huruf saja. Perhatikan sintaks perintahnya:

```
grep '^......\?.\?$' /usr/share/dict/words
```

Di **grep**, karakter meta ? justru *harus* diescape agar berfungsi. Jika tidak diescape, maka akan dianggap karakter literal. Kelakuan ini terbalik dengan rata-rata regex di tempat lain (PHP, Perl, Python, dan lain sebagainya).

9-1 Nomor Telepon

Masalah

Anda ingin mengecek apakah sebuah string merupakan nomor telepon.

Pola regex

```
# A. agak ketat: 021-5401234 atau 021-540-2312
/^\d{3,4}-\d{3}-?\d{4}$/

# B. liberal
/^[\d- ]+$/

# C. juga mengizinkan (area)-nomor
/^(\(\d+\)[- ]?)?[\d- ]+$/

# D. kode internasional
/^((\+\d+)[-\s]*)? # kode negara, opsional. misalnya: +62-
  (\(\d+\)[-\s]*)? # kode wilayah, opsional. misalnya: (021)-
  [\d-\s]+ # nomor. misalnya: 540-1234
```

```
$/x

# E. format internasional yang standar: +62215269311
/^\+                    # diawali +
  [1-9][0-9]{0,2}       # kode negara 1-999
  [1-9][0-9]{0,3}       # kode wilayah 1-9999
  [1-9][0-9]{1,7}       # maks 8 digit nomor lokal, tidak boleh
kepala 0
 $/x;
```

Diskusi

Mengecek nomor telepon bisa menjadi latihan regex yang baik. Ada berbagai variasi sintaks nomor telepon yang dipakai orang. Sebagai latihan, mungkin Anda bisa menambahkan kode untuk membatasi jumlah angka yang wajar, melarang format seperti "-1-2-3-4-", dan sebagainya.

Format standar internasional seperti ditunjuk oleh regex E mulai banyak dipakai secara internal oleh software untuk mencatat nomor telepon.

Catatan: kode negara, kode wilayah, dan nomor lokal tidak dapat diawali 0 karena biasanya 0 ini merupakan kode khusus untuk mengakses fungsi tertentu (misalnya: menekan 0-XXX berarti mengakses interlokal dengan kode wilayah XXX; menekan 017-XXX berarti mengakses SLI via Telkom ke negara dengan kode internasional XXX; angka 0 juga dipakai di sistem PABX untuk mengakses line eksternal, dan lain sebagainya).

9-2 Kode Pos

Masalah

Anda ingin menguji apakah sebuah string sintaksnya seperti kode pos.

Pola regex

```
# A. kode pos Indonesia
/^\d{5}$/

# B. kode pos Amerika 9 digit
/^\d{5}-\d{4}$/

# C. kode pos Inggris
/^[A-Za-z]{1,2}\d{1,2}([A-Za-z])?\s?[\d][A-Za-z]{2}$/
```



```
/^[A-Z][A-Z]?\d\d?[A-Z]?\s?\d[A-Z][A-Z]$/

# D. kode pos
```

Diskusi

Perlu diingat bahwa format kode pos berbeda-beda di setiap negara. Ada yang hanya 5 digit seperti Indonesia (aneh juga, karena sebetulnya Indonesia amat luas), hingga 12-14 digit. Ada yang hanya mengandung angka, ada juga yang dapat mengandung huruf. Tentu saja, tidak semua angka merupakan kode pos yang dikenali. Untuk benar-benar mengecek apakah kode pos dikenali, Anda memerlukan database kode pos. Untuk kodepos Indonesia, Anda bisa bertanya pada PT Pos (http://www.posindonesia.co.id) atau datang ke kantor pos setempat.

9-3 Nomor ISBN

Masalah

Anda ingin menguji apakah sebuah string menyerupai nomor ISBN.

Pola regex

```
# hanya mengecek sintaks, tanpa mengecek check digit
/^[0-9]{9}[0-9Xx]$/
```

Contoh kode Perl

```
function is_valid_isbn {
  my $isbn = shift;
  $isbn =~ s/[^0-9Xx]//g;
  return 0 unless /^(\d)(\d)(\d)(\d)(\d)(\d)(\d)(\d)(\d)([0-
9Xx])$/;
  my $chk = $10 eq 'X' || $10 eq 'x' ? 10:$10;
  return 0 unless
    ($1*10+$2*9+$3*8+$4*7+$5*6+$6*5+$7*4+$8*3+$9*2+$chk % 11 ==
0);
  return 1;
}
```

Diskusi

ISBN adalah nomor global untuk sebuah judul buku, terdiri dari 9 digit diikuti sebuah check digit. Check digit dapat berupa 0 hingga 9 atau X yang berarti 10. Cara menghitung check digit diperlihatkan di contoh kode Perl.

9-4 Tanggal/waktu

Masalah

Anda ingin menguji apakah sebuah string merupakan tanggal/waktu yang valid, dan ingin mengambil nilainya.

Pola regex

```
# A. tanggal numerik: dd-mm-yyyy atau dd-mm-yy
m#^(\d\d?)[-/ ](\d\d?)[-/ ](\d\d(?:\d\d)?)$#

# B. jam: HH:MM:SS
/^\d\d?:\d\d?:\d\d?$/

# C. bulan bahasa indonesia, menangani beberapa variasi ejaan,
notasi singkat
m#^(\d\d?)[-/\s]
    (jan|[fp]eb|mar|apr|mei|jun|jul|agu|sep|no[pv]|des)[-/\s]
    (\d\d(?:\d\d)?)$#xi

# D. bulan bahasa indonesia, notasi panjang
m#^(\d\d?)[-/\s]
    (januari|[fp]ebruari|maret|april|mei|juni|juli|
     agustus|september|no[pv]ember|desember)[-/\s]
    (\d\d(?:\d\d)?)$#xi

# E. diprefiks nama hari, pendek maupun panjang
m#^(sen(?:in)|sel(?:asa)?|rabu?|kam(?:is)?|jum(?:'?at)?|sa[bp]tu|min(?:ggu)?)
    (?:,?\s)
    (\d\d?)[-/\s]
    (\d\d?)[-/\s]
    (\d\d(?:\d\d)?)$#x
```

Diskusi

Salah satu kompleksitas yang dibuat manusia yang berpengaruh pada pemrograman komputer adalah sistem waktu dan penanggalan. Manusia banyak membuat sistem penanggalan yang beraneka ragam dan kadang direvisi beberapa kali; menggunakan satuan yang tidak bulat/bervariasi (misalnyaalnya: 1 bulan tidak selalu 30 hari; ada leap second, leap year, daylight savings time); digunakan kata bahasa manusia untuk menyebut bulan dan hari (sehingga ada perbedaan bahasa, ejaan, singkatan, dan lain sebagainya).

Komputer sendiri umumnya menyimpan tanggal/waktu dalam format timestamp berupa bilangan integer 32-bit/64-bit. Format seperti ini amat nyaman dan ringkas bagi komputer dalam melakukan aritmetika. Tapi manakala kita harus berurusan dengan notasi manusia, kita harus menggunakan berbagai pola regex untuk mengubahnya menjadi timestamp.

Pola di resep ini hanya memberi contoh beberapa jenis variasi notasi saja. Anda dapat melihat pada modul Perl `Date::Manip` atau `strtotime()` untuk kemampuan parsing yang lebih lengkap. Sayangnya, rata-rata modul ini dibuat untuk teks bahasa Inggris. Namun berbekal kemampuan regex, rasanya tidak sulit bagi Anda membuat modul/fungsi serupa untuk bahasa Indonesia.