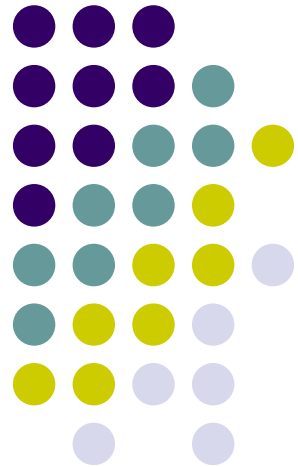
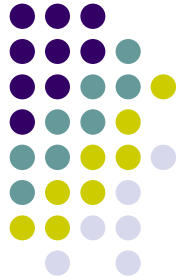
Teknik Kompiler 3

oleh: **antonius rachmat c, s.kom**



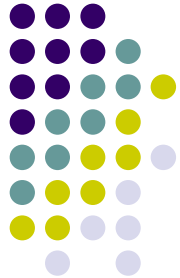
REGULAR EXPRESSION

sumber buku: **“Mastering Regular Expressions, 2nd EditionBy Jeffrey E. F. Friedl, O’Reilly, July 2002”**



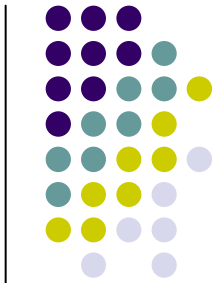
- RE adalah bahasa, atau bahkan mirip bahasa pemrograman mini untuk mendeskripsikan dan memparsing string atau teks.
- RE merupakan notasi (*pattern notation*) yang dapat digunakan untuk mengolah teks (*describe and parse text*).
- RE sering digunakan untuk “search and replace”.

Story of the real situation



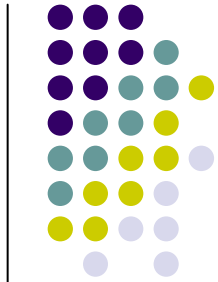
- Seorang programmer harus membuat sebuah tool, yang dapat mengecek kata-kata yang dobel seperti "kemarin-kemarin", Tugas programmer itu adalah membuat program yang :
  - Menerima banyak file untuk diperiksa, melaporkan setiap baris yang memiliki kata yang dobel, di-*highlight*, dan nama file akan muncul pada setiap file yang dilaporkan
  - Mampu memeriksa pada seluruh baris, bahkan untuk kata yang ada diakhir baris dan kata yang sama berikutnya ditemukan pada awal baris.
  - Mampu mencari kata yang dobel secara *incase-sensitive*, seperti “Lagi lagi”, dan walaupun dipisahkan oleh banyak white-space karakter sekalipun.
  - Mampu mencari kata dobel yang ada di teks HTML atau memiliki karakter tag. Seperti contoh ini: '…it is <B>very</B> very important'.
- Sulit? That is the real problems!
- Dengan RE kita dapat meng-*highlight* kata yang ulang, membuang setiap baris yang tidak ada kata yang ulangnya, dan kita bisa membuat setiap baris menampilkan nama filenya.

The other side of RE



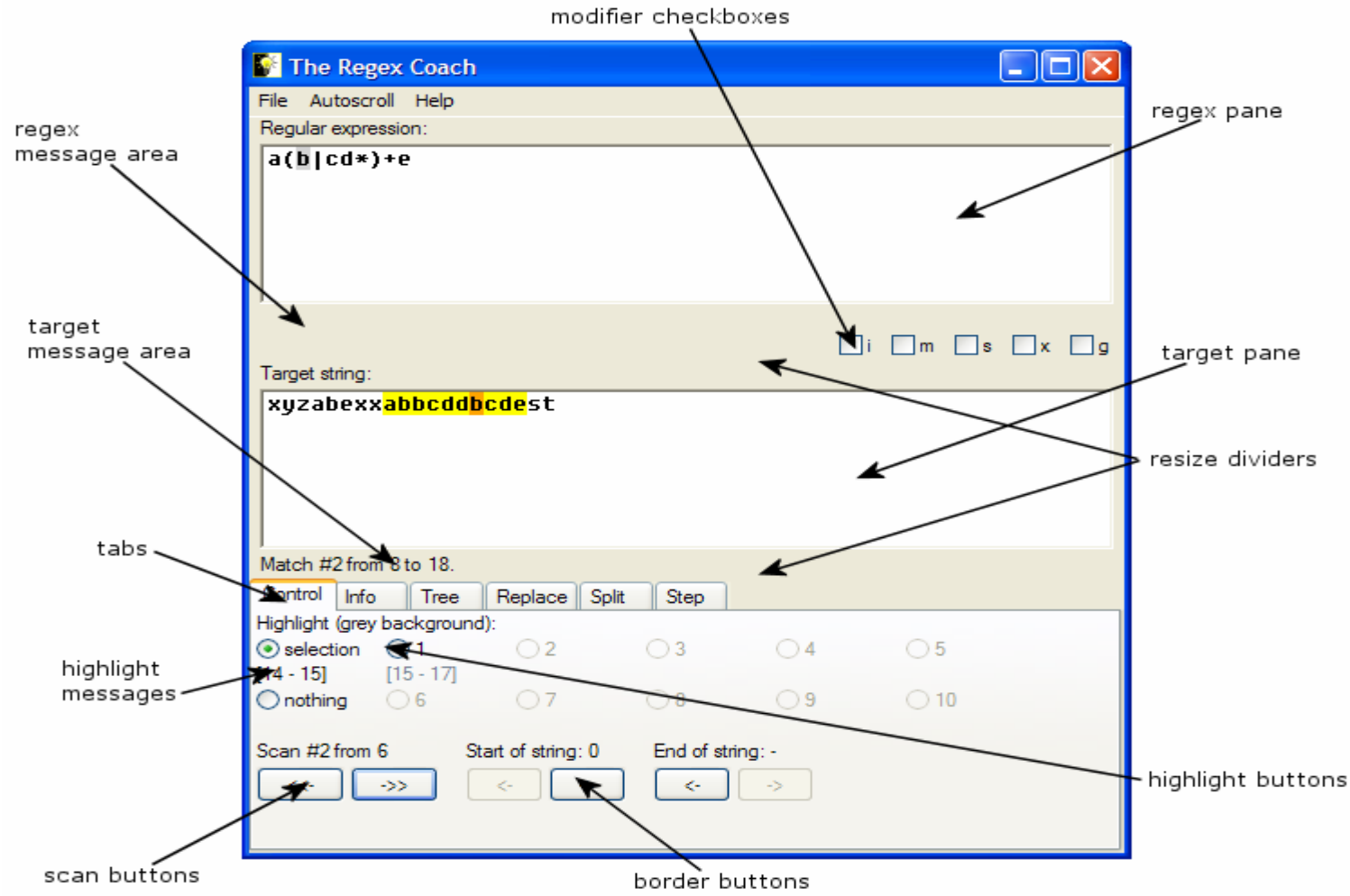
- Regular Expression sering disebut Regex atau RE
- Misal RE akan bersifat “*match in*”. Misal RE “a” akan *match in* “cat”, bukan *match* dengan “cat”
- Ada banyak sekali varian RE (RE flavor). Hal ini disebabkan oleh dukungan metakarakter dan artinya.
- Character di dunia ini luar biasa banyak, maka RE harus memperhatikan character encoding-nya.
- RE ditemukan pada tahun 1940an awal oleh dua neurophysiologist, Warren McCulloch dan Walter Pitts pada saat membuat model neuron (syaraf).
- Kemudian model tersebut dibuat aljabarnya oleh seorang matematika Stephen Kleene dan diberi nama “regular expression”.
- Tahun 1968 = muncul buku “Regular Expression Search Algorithm” dari IBM

Tool RE

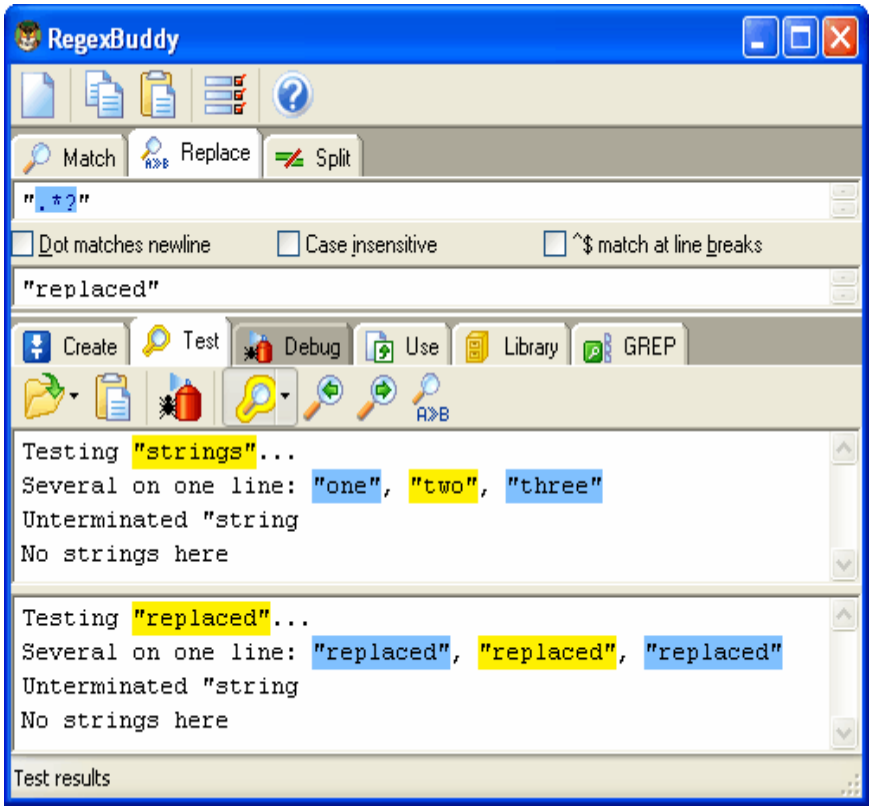


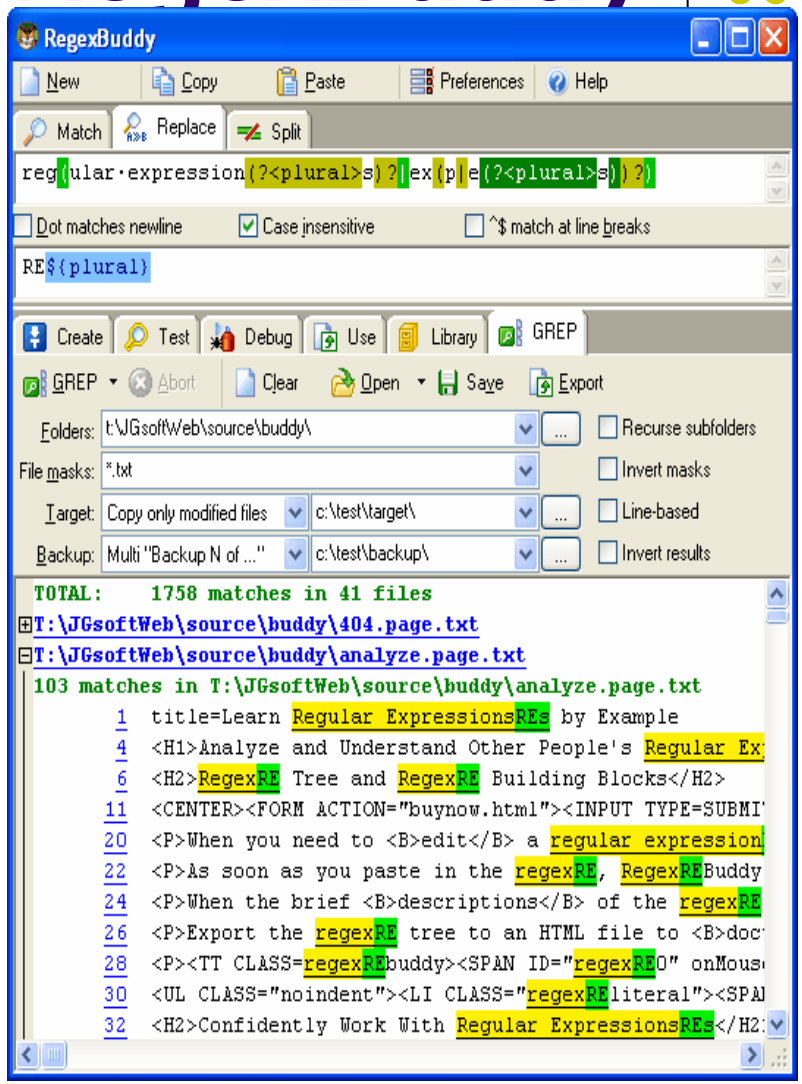
- Ditemui di grep, QED, AWK, emacs, vi
- Salah satu tool yang dapat digunakan untuk belajar RE adalah http://weitz.de/files/regex-coach.exe (*Regex Coach*) yang dibuat oleh Dr. Edmund Weitz untuk Windows atau http://weitz.de/files/regex-coach.tgz untuk Linux. Program ini bersifat *donationware*! ☺
- Notepad++, Notepad2, OpenOffice, RegexBuddy, PowerGREP dan lain-lain.
- Regexstudio.com
- Kodos.sourceforge.net

Capture Regex Coach

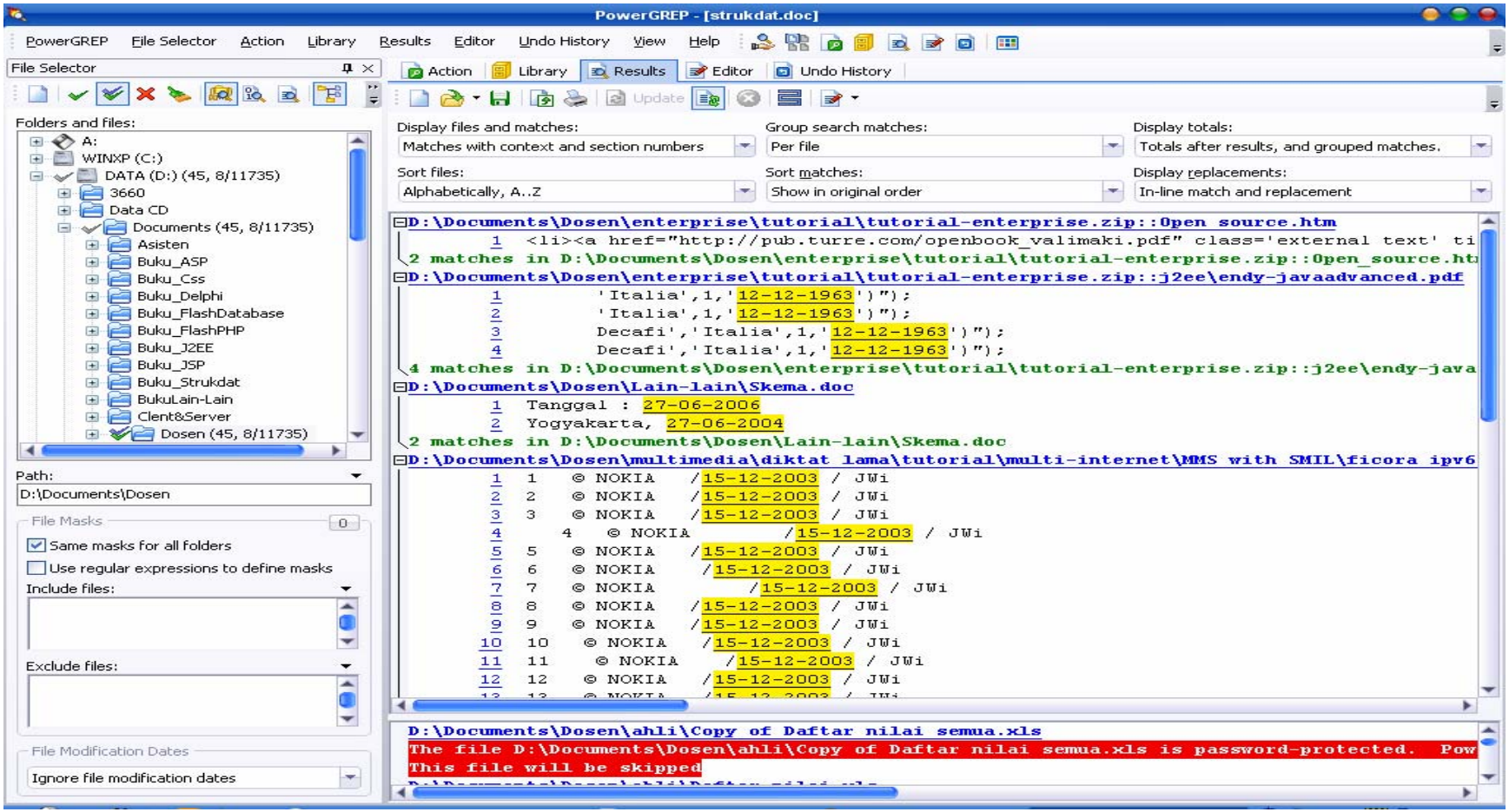


Capture Program RegexBuddy

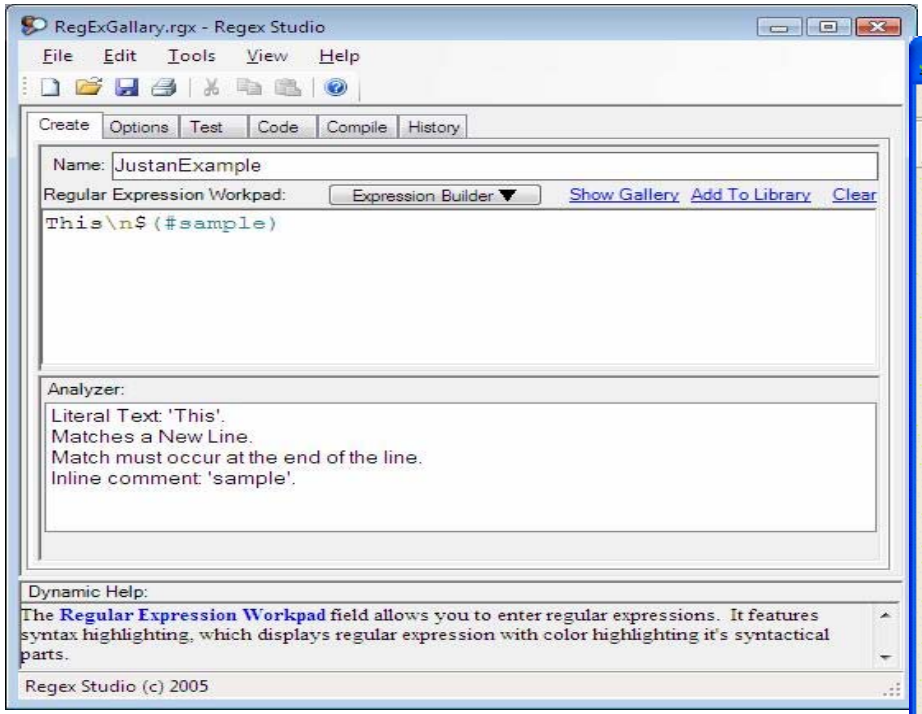


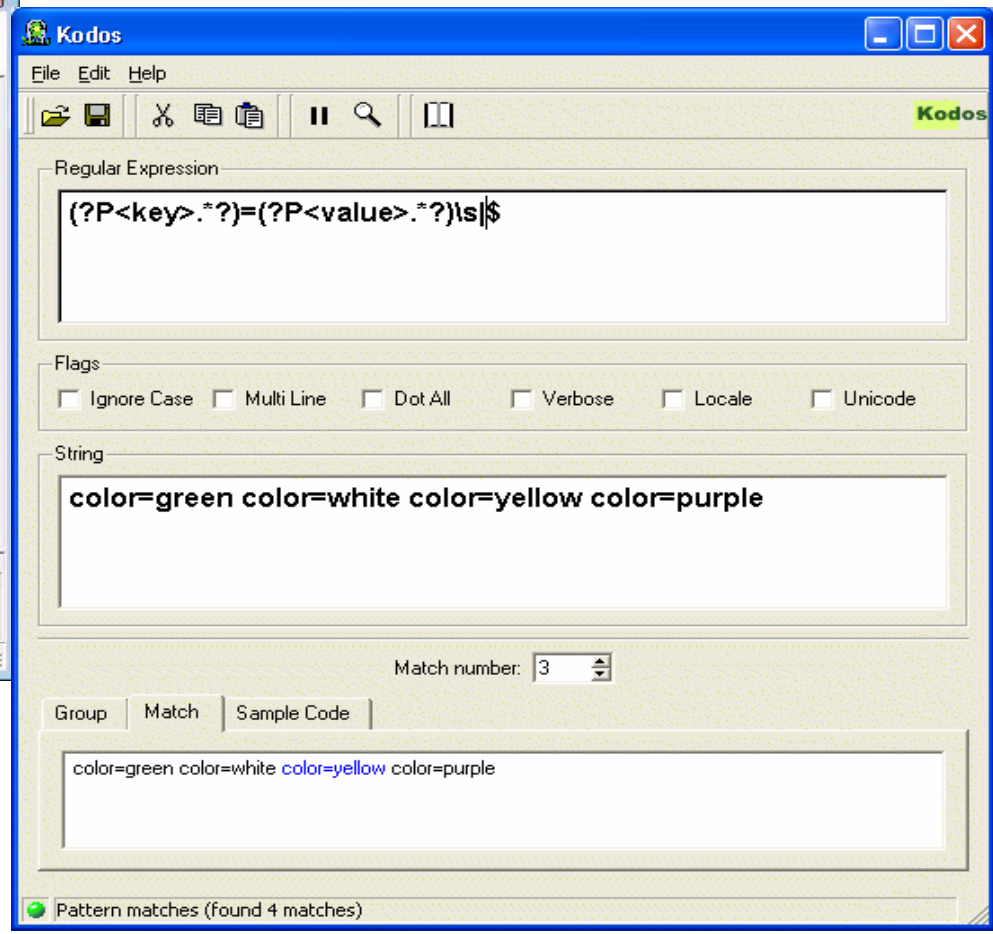


Capture PowerGREP

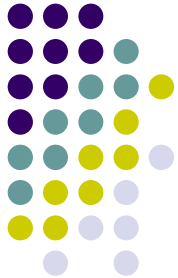


Regexstudio (.NET) dan Kodos



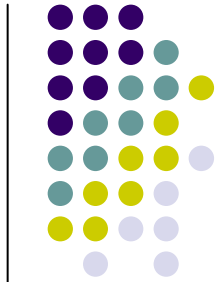


Analogi RE



- RE dapat dianalogikan dengan berbagai function pengolah string pada bahasa pemrograman yang belum mendukung RE. Misalnya strcmp(), length(), mid(), trim(), substr(), pos(), strstr() dan lain-lain.
- RE juga bisa dianalogikan/mirip dengan fungsi-fungsi *WildCard* pada DOS/UNIX untuk pengelolaan file. Ingat : *.txt, a*.t?t, atau di grep/sed/awk di Linux?
- Hati-hati jika salah menggunakan RE, misal mencari kata “**cat**” maka “va**cat**ion” akan ikut ditemukan!

remember DOS?



- DOS had the * character as a wildcard. If you said

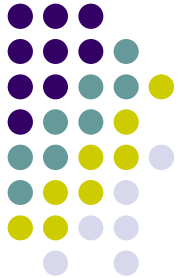
  DIR *.EXE

  - It would list all the files ending with .EXE
- Thus the * wildcard would mean “all characters except the dot”
- Similarly, you could say

  DEL *.*

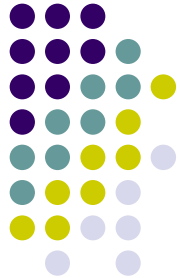
  - to delete all your files

Kelebihan RE



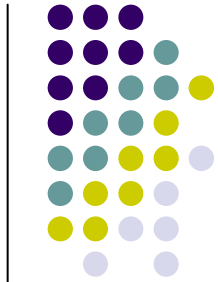
- Sangat ampuh untuk mengelola dan mencocokan file teks atau string.
- Sangat ringkas, karena sintaks RE sangat “sederhana” untuk melakukan hal yang “besar”.
- RE cepat, karena menghindari kita melakukan pemeriksaan manual dan RE cepat dalam mencocokan pola-polanya.
  - Misalnya untuk mencari suatu string “Saya”, “saya” di dalam file dengan 10000 baris dan akan direplace menjadi “Dia”.

Aturan RE



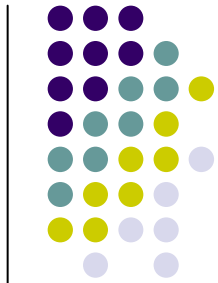
- Leftmost – Rightmost
- Quantifier is greedy

Bagian Utama RE



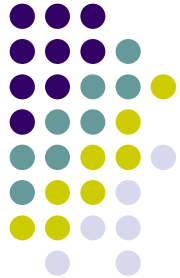
- *Meta character* : yaitu karakter-karakter yang memiliki arti khusus di RE dan mewakili sekelompok karakter lain atau pola khusus tertentu.
  - Yang termasuk: *, . (titik), [,], (,), ^, $, +, dan lain-lain.
- *Literal character* : yaitu karakter-karakter biasa (yang diterjemahkan apa adanya) yang tidak memiliki arti khusus.

Lebih lanjut tentang RE



- Setiap tool yang mengimplementasikan / menggunakan RE pasti menggunakan **library RE** dan **mesin RE**.
- **Mesin RE** adalah program yang menerima string pola RE dan mengkompile-nya menjadi RE tree yang berupa mesin state.
- Sebuah RE harus dikompile dulu agar dapat digunakan.
- Mesin RE akan berusaha mencocokkan semua kemungkinan dari RE untuk string yang dicocokkan.
- Jika mesin RE mencapai state akhir maka dinyatakan bahwa pola RE dan string sesuai, tapi jika tidak maka mesin RE akan mencoba melakukan *backtracking* semua kemungkinan lain sampai semua kemungkinan dicoba, jika sudah semua dan masih tetap belum mencapai state akhir, maka pola dnyatakan tidak cocok.

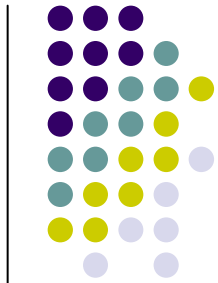
Library RE



- Library RE yang terkenal adalah library PERL, karena sangat cepat dan optimal.
- Aturan RE yang sekarang mengikuti aturan salah satu berikut ini:
  - Aturan Library **POSIX**. Dibuat oleh Henry Spencer. POSIX membagi RE menjadi dua bagian:

    Standar RE dan Advanced RE.
  - Aturan Library **GNU Regex**. Dibuat dengan bahasa C, dimuat di dalam glibc.
  - Aturan Library **PCRE**. Dibuat oleh Philip Hazel

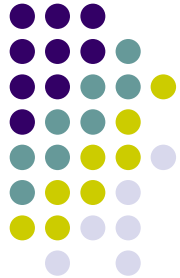
Bahasa Pemrograman Pendukung RE



- Java
  - Misalnya Java dengan:

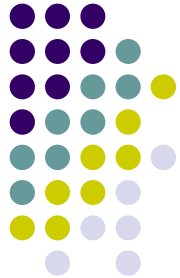
    ```
    import java.util.regex.*
    import org.apache.regexp.RE.*
    ```
- PHP mendukung RE POSIX (misalnya : `ereg()`) dan RE PCRE (misalnya `preg_match()`).
- Python (gunakan `re`), .NET (gunakan `System.Text.RegularExpression`)
- Namun ada yang belum mendukung RE misalnya: Delphi, Pascal, dan C.
  - Untuk Delphi bisa dibantu dengan menggunakan komponen lain yaitu TRegExpr (http://regexpstudio.com/TRegExpr/TRegExpr.html/)

KARAKTER META DASAR



- Pemilihan (Alternation)
  - Menggunakan karakter "**|**" yang mensimbolkan pemilihan.
  - Sintaks: A1|A2|A3 …. Dan seterusnya
  - Dimana A1, A2, A3,… adalah sub pola.
  - Karakter | dibaca "atau", "salah satu cocok".
  - Contoh pola: "http|ftp|smtp" berarti salah satu string http atau ftp atau smtp cocok. Akan cocok dengan kalimat: http://www.ukdw.ac.id atau ftp://ftp.students.ukdw.ac.id
  - Diagram state?

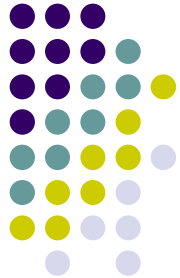
KARAKTER META DASAR (2)



- Pengelompokkan (Grouping)
  - Menggunakan karakter “(“ dan “)”.
  - Contoh:

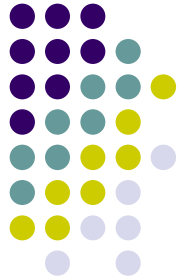
    Mahasiswa teknik (informatika|sistem informasi|arsitektur) UKDW.
  - Berarti akan cocok dengan “Mahasiswa teknik informatika UKDW” atau “Mahasiswa teknik arsitektur UKDW”, tapi tidak cocok dengan kalimat “Mahasiswa teknik ekonomi UKDW”.
  - Diagram state?

KARAKTER META DASAR (3)



- Karakter apa saja (*any character*)
  - Menggunakan karakter “.” (titik).
  - Hati-hati dengan karakter titik yang “sebenarnya”
  - Hati-hati juga dalam analogi Wildcard file!
  - Contoh : Anton... Berarti kalimat Antonius akan cocok, Antonies, tapi tidak akan cocok dengan Anton, Antony. Karena tanda titiknya ada tiga buah, maka akan berarti kalimat Anton harus diikuti dengan 3 karakter apa saja.
  - Diagram State?
  - Contoh lain: 07.04.76
  - Masukkah string berikut? “lottery numbers: 19 207304 7639”

KARAKTER META DASAR (4)



- Karakter kelas (*character class/character sets*)
  - Kumpulan karakter digunakan tanda “[“ dan “]” yang digunakan mendefiniskan sekumpulan karakter yang cocok.
  - Mirip dengan pemilihan, hanya saja kelas karakter hanya berlaku untuk 1 karakter tunggal.
  - Di dalam karakter kelas, dikenal karakter range (-) dan karakter negasi (^).

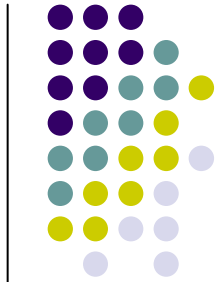
Karakter Kelas

- Contoh :

Pola	yang cocok	yang tidak cocok
[abc]	a	d
[a-zA-Z]	B	1
[^1-3]	4	2
[aA][bB][cC]	AbC	AcB

- Buatlah pola regex untuk menerima 4 huruf konsonan dan setelahnya 1 karakter vokal.

Attention

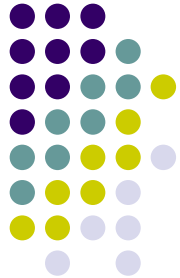


- Karakter-karakter seperti |, . (titik), () yang merupakan karakter meta akan menjadi karakter literal jika berada di dalam karakter kelas. Contoh : [(a|b)] akan menjadi pola karakter (,a,|,b) bukan karakter salah satu dari a atau b. Titik di dalam kelas karakter akan menjadi karakter titik sebagaimana mestinya bukan menjadi karakter apa saja.
- Karakter minus – dan pangkat ^ akan menjadi karakter meta di kelas karakter. Tanda – berarti tanda range (rentang) dan tanda ^ berarti karakter negasi yang berarti bukan (not)

Attention (2)

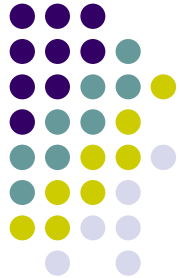
- Tanda minus – jika berada di kelas karakter bagian akhir (terletak dibelakang) akan menjadi karakter literal, yang mensimbolkan karakter minus itu sendiri bukan karakter rentang lagi. Contoh : [0-9-] berarti karakter nol sampai sembilan atau karakter minus. Misal string 9 atau – akan cocok dengan pola diatas.

KARAKTER META DASAR (5)



- Karakter Jangkar/*Anchor*
  - Tanda ^ Berarti awal baris atau awal string.
  - Tanda $ berarti akhir string.
  - Akhir/awal baris atau akhir/awal string bergantung pada modifier atau mode operasinya.
  - Karakter ^ dan $ tidak mewakili suatu karakter apapun tapi mewakili posisi.
  - Contoh : ^hari berarti dibaca kalimat atau baris yang diawali dengan deretan huruf hari. Contoh yang cocok : “hari minggu”, “harimau”.
  - Tanpa tanda jangkar maka string “seharian” atau “berharian” akan cocok.
  - Sedangkan hari$ berarti dibaca kalimat atau baris yang diakhiri sederetan huruf “hari”.
  - ^$ berarti baris kosong.

KARAKTER META DASAR (6)



- Contoh lain :

  % egrep 'q[^u]' word.list
  Iraqi
  Iraqian
  miqra
  qasida
  qintar
  qoph
  zaqqum
  %

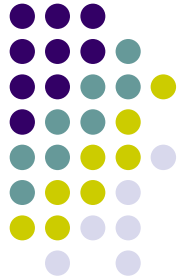
- Yang belum muncul Iraq dan Qantas. Kenapa?

  % egrep '^(From|SubjectlDate): ' mailbox

  Hasil:

  From: elvis@tabloid.org (The King)
  Subject: be seein' ya around
  Date: Thu, 31 Oct 04 11:04:13
  From: The Prez <president@whitehouse.gov>
  Date: Tue, 5 Nov 2002 8:36:24
  Subject: now, about your vote...

KARAKTER META DASAR (7)



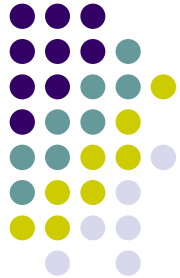
- **Quantifier**
  - Menggunakan karakter { }.
  - Sintaks : P{m,n} atau P{m,} atau P{m} dimana P adalah pola suatu regex dan m & n adalah integer bilangan cacah yang merupakan pembatas jumlah keluaran pola.
  - Contoh: [0-9]{3} berarti karakter antara 0-9 sebanyak tepat 3 kali muncul.
  - [a-z]{2,3} berarti karakter antara a-z minimal 2 maksimal 3 kali muncul.
  - Cobalah membuat RE untuk memvalidasi IPv4 misalnya 192.168.1.2

KARAKTER META DASAR (8)

- **Optional (*zero or one*)**
  - Menggunakan karakter ? yang sama artinya dengan {0,1}
  - Hati-hati bedanya dengan WildCard.

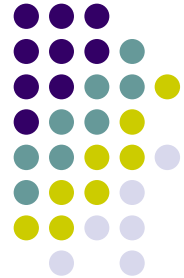
    Pada WildCard tanda ? bisa digantikan dengan karakter . pada regex yang artinya 1 karakter apa saja.
  - Contoh: `colou?r`
  - Lihat kalimat “July fourth”, padahal orang kadang menulis July atau Jul dan fourth atau 4^{th} atau 4 saja.
  - Bagaimana RE nya ? `(July|Jul) (fourth|4th|4).`
  - Kita bisa memperpendeknya menjadi `(July?) (fourth|4(th)?)`

KARAKTER META DASAR (9)

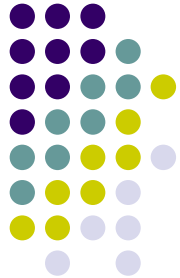


- **Zero or more**
  - Menggunakan karakter * yang sama artinya dengan {0,}.
  - Arti * pada regex berbeda dengan * pada WildCard.
  - Tanda * pada WildCard akan menghasilkan semua karakter, sedangkan pada regex akan menjadi .* yang artinya 0 atau lebih karakter apa saja.
- **One or more**
  - Menggunakan tanda + yang artinya sama dengan {1,}
  - Cobalah membuat regex email sederhana.

KARAKTER META DASAR (10)



- **Kelas karakter digit (angka)**
  - Terdapat beberapa simbol khusus untuk melambangkan kelas karakter dengan tujuan mempersingkat penulisan.
  - Sintaks:
    - \d yang sama artinya dengan [0-9]
    - \D yang sama artinya dengan [^0-9]
  - Apa artinya dengan \Dd+\D? Mana yang cocok? X123X? atau 123X? atau “ 123 “?
- **Kelas karakter huruf (alphanumerik)**
  - Menggunakan sintaks : \w dan \W sebagai komplemennya.
  - \w artinya sama dengan [0-9A-Za-z_] dan \W sama artinya dengan [^0-9A-Za-z_].
  - Bagaimana mendeteksi sebuah kata?



- TO BE CONTINUED tommorrow!