Bab 1

Pendahuluan

Pada bab ini akan diperkenalkan regex,
serta gambaran apa saja yang bisa dilakukan oleh regex.
Bab ini juga menjadi pengantar bagi bab-bab selanjutnya.

Apa Itu Regex?

Regular expression (ekspresi regular, regexp, regex, RE) adalah sebuah bahasa mini untuk mendeskripsikan string atau teks. Regex dapat dipakai untuk mencocokkan sebuah string dengan sebuah pola. Analogikan hal ini dengan fungsi string strcmp(), strcasecmp(), dan strpos(). Tapi pencocokan string dengan regex jauh lebih ampuh. Selain menguji kecocokan atau mencari substring dalam string, regex juga dapat dipakai mengekstrak string dari teks. Analogikan hal ini dengan fungsi string left(), right(), mid(), atau substr(). Namun versi regex jauh lebih powerful. Terakhir, regex juga bisa dipakai untuk membelah dan mensubstitusi substring dengan string lain.

Regex sebetulnya telah diciptakan lama dan telah dipakai oleh berbagai program dan bahasa pemrograman dalam satu bentuk atau lainnya. Meskipun mula-mula hanya dijumpai dalam beberapa versi **grep** di Unix dan juga di DOS, kini semua orang dapat menjumpai dan menggunakan regex di berbagai *bahasa pemrograman*, berbagai *OS*, dan berbagai *aplikasi*.

Misalnya, di Microsoft Word, bisa dijumpai Special Characters seperti Any Character, Any Digit, Any Letter. Karakter-karakter spesial ini mirip dengan regex, yaitu untuk mencocokkan sebuah pola string. Para pengguna shell Unix tentu akrab dengan wildcard dan perintah-perintah yang menggunakan regex. Bukan saja **grep** melainkan juga **awk** dan **sed**. Wildcard bisa disebut sebagai "saudara tua" regex. Dengan wildcard, kita dapat menspesifikasikan sekelompok file yang memiliki nama dengan pola tertentu. Misalnya, *.txt berarti semua file yang berakhiran .txt atau [abc]* berarti semua file yang namanya diawali huruf a, b, atau c. Sementara dengan grep, Anda bisa mencocokkan baris-baris teks yang memiliki pola tertentu, misalnya grep '^From ' *.mbox untuk mencari dan menampilkan baris-baris yang diawali dengan lima karakter ("From "). Di Perl, Python, atau PHP, Anda akan menemukan fasilitas regex. Di editor seperti vi dan joe pun regex ada. Di ASP, .NET, Java, dan Javascript pun kelas regex bisa ditemui.

Mengapa Belajar Regex?

Keberadaan regex di berbagai tool dan bahasa pemrograman membuktikan bahwa regex sangat berguna dalam mengolah teks. Kenapa demikian? Pertama, karena regex sangat **ampuh**. Anda bisa membuat pola regex untuk mencocokkan dan mengekstrak pola string mulai dari angka, nomor telepon, kata, URL, hingga pola definisikan Anda sendiri atau custom. Dengan regex, Anda bisa mengambil potongan string yang diinginkan dari file log, dokumen HTML, bahkan file biner sekalipun. Saat fungsi string atau fasilitas search and replace editor teks sudah

angkat tangan, Anda bisa berharap pada regex. Kedua, regex amat **ringkas**. Regex dapat menggantikan belasan hingga puluhan baris kode program hanya dengan sebuah pola. Misalnya, untuk mencocokkan apakah sebuah teks mengandung alamat email atau tidak, hanya dengan sebuah pola yang terdiri dari beberapa puluh karakter saja. Karena ringkas, regex jadi lebih cepat ditulis ketimbang baris demi baris kode program. Ketiga, regex **cepat**. Seringkali dengan menggunakan regex, Anda bisa menghindari melakukan puluhan atau ratusan perbandingan string secara manual. Library regex yang ada sekarang ini, terutama library atau mesin regex di Perl, rata-rata sudah teroptimasi sehingga pola yang kompleks sekalipun bisa dijalankan dengan sangat cepat.

Dengan ketiga manfaat yang diberikan regex tersebut, Anda bisa mengolah teks secara lebih mudah dan cepat. Baik dari sisi kemudahan dan kecepatan membuat pola maupun kecepatan eksekusi.

Seperti Apa Bentuk Regex?

Pola regex sebetulnya dinyatakan dengan string biasa. Misalnya, pola regex aku akan cocok dengan semua teks yang mengandung potongan string aku, seperti aku, kaku, laku, akur, dan takut (tapi tentu saja tidak cocok dengan string kamu, uka, atau angkut; karena tidak ada aku di sana). Dalam hal ini pencocokan regex tidak ada bedanya dengan fungsi strpos(), yaitu mencari substring dalam string. Dan sama halnya juga dengan pencocokan string biasa yang dapat dibuat case sensitive (membedakan huruf besar-kecil) atau case insensitive (tidak membedakan huruf besar-kecil), pola regex aku juga dapat dibuat cocok dengan Aku atau AKU atau tidak, bergantung pada apakah kita ingin pencocokan dilakukan secara case sensitive atau insensitive.

Yang membedakan regex dari string biasa adalah terdapatnya karakter-karakter khusus yang disebut *karakter meta* (metacharacters). Karakter-karakter ini tidak akan dicocokkan secara literal dengan karakter itu sendiri, tapi mewakili sekelompok karakter lain atau pola khusus tertentu. Analogikan dengan wildcard, di mana terdapat dua karakter meta utama yaitu * (asterisk) dan ? (tanda tanya).

Jika menyebutkan wildcard (*.php), kita tentu saja tidak bermaksud mencari nama file yang benar-benar diawali dengan karakter asterisk. Malah kenyataannya asterisk tidak diperbolehkan sebagai bagian dari nama file yang valid di DOS atau Windows. *.php berarti semua file yang berakhiran PHP, seperti 1.php, dua.php, tiga dan empat.php, dan sebagainya. Asterisk adalah karakter meta, karena saat disebutkan asterisk tidak berarti karakter asterisk itu sendiri melainkan mewakili sebuah pola yaitu "nol atau lebih karakter apa saja."

Sementara ? adalah karakter meta yang berarti "satu karakter apa saja." Contoh wildcard yang mengandung ?: gambar??.jpg yang berarti semua file .jpg yang diawali string gambar dan diikuti dua karakter apa saja, misalnya gambar01.jpg, gambar09.jpg, gambar-5.jpg, atau gambaran.jpg.

Regex, sama seperti wildcard, mengenali karakter-karakter meta. Karakter ? dan * juga merupakan karakter meta di regex meskipun artinya sedikit berbeda. Bedanya dengan wildcard, regex mengenal lebih banyak karakter meta. Di antaranya [] (kurung siku), + (plus), ^ (caret atau pangkat), $ (dolar), \s (backslash s), . (titik), () (kurung), dan sebagainya. Kita akan membahas karakter-karakter meta ini di Bab 2.

Dengan adanya karakter-karakter meta ini, kemampun regex menjadi lebih ampuh daripada wildcard yang terbatas. Kita dapat menyatakan semua pola wildcard dalam versi regex-nya, tapi tidak sebaliknya. Sebagai contoh, pola wildcard *.php di regex kita tulis sebagai .*\.php dan pola wildcard gambar??.jpg dalam bahasa regex ditulis sebagai gambar..\.jpg atau gambar.{2}\.jpg. Sementara pola regex yang lebih rumit seperti (\w+)-\1\.jpg atau file .jpg yang memiliki kata ulang, seperti kupu-kupu.jpg, mata-mata.jpg, atau kura-kura.jpg tidak bisa ditulis dalam bentuk atau pola wildcard. Hal ini dikarenakan wildcard tidak mengenal karakter meta backtrack seperti \1, apalagi wildcard DOS atau Windows.
Wildcard di shell Unix sebetulnya lebih ampuh, mengenal juga beberapa karakter meta lain seperti [] dan {}).

Library dan Mesin Regex

Bahasa pemrograman atau editor teks yang memiliki fasilitas regex biasanya memiliki atau menggunakan *library regex*. Library inilah yang mengimplementasikan *mesin regex*. Mesin regex adalah software atau kode program yang menerima pola regex dalam bentuk string dan meng-*compile*-nya. Sebuah pola regex harus di-compile dulu sebelum dapat digunakan. Dalam kompilasi, pola regex mentah berbentuk string diterjemahkan menjadi struktur tree yang merupakan mesin state (state machine). Setelah dikompilasi, pola regex dapat digunakan untuk pencocokan. String yang ingin dicocokkan kita berikan kepada mesin regex. Kemudian mesin regex akan menjalankan atau menelusuri mesin state regex hasil kompilasi dan mengujinya terhadap string.
Jika mesin regex berhasil mencapai akhir atau ujung mesin state, maka dikatakan string cocok dengan regex yang bersangkutan. Jika gagal pada sebuah tahap di mesin state, mesin regex dapat melakukan backtrack atau cara lain agar sebuah string dapat cocok. Jika mentok, barulah mesin regex menyerah dan string dikatakan tidak cocok dengan regex tersebut. Kita akan melihat ilustrasi bagaimana mesin regex bekerja di Bab 2.

Jadi, mesin regex berperan dalam *kompilasi* dan *eksekusi* bahasa regex. Analogikan ini dengan bahasa pemrograman biasa. Sebuah program sumber mula-mula kita kompilasi dulu menjadi kode mesin atau kode bytecode, lalu prosesor atau virtual machine mengeksekusi kode hasil kompilasi ini. Dalam hal ini mesin regex dapat disamakan seperti kompiler dan virtual machine.

Bahasa pemrograman seperti Perl memiliki dan mengembangkan library regex-nya sendiri. Regex Perl dikenal sebagai salah satu library yang memiliki mesin regex tercepat dan terampuh di dunia. Library regex Perl telah dioptimasi selama bertahun-tahun. Bahasa pemrograman lain seperti Ruby dan Python juga mengembangkan sendiri library regex-nya, tapi berpatokan pada regex Perl sebagai standarnya. Dan bahasa lain, seperti PHP atau rata-rata aplikasi lain, menggunakan library regex yang telah ada. Seperti regex GNU yang ada di **glibc** -library standar C, yang digunakan nyaris semua program dalam bahasa C-, library regex POSIX yang ditulis oleh Henry Spencer, dan **PCRE** (Perl-compatible Regular Expression library, library yang mengimplementasikan mayoritas sintaks regex Perl). Contohnya, untuk fungsi regex eregi() atau ereg_replace(), PHP menggunakan library regex POSIX buatan Henry Spencer. Sementara untuk fungsi-fungsi regex ala Perl preg_match(), preg_replace(), dan preg_split(), PHP menggunakan library PCRE yang merupakan karya Philip Hazel (perlu dicatat bahwa Philip Hazel juga menulis software MTA **Exim**; Exim banyak memanfaatkan regex di file konfigurasinya—fasilitas regex di Exim tentu saja diimplementasi menggunakan library PCRE).

Flavor Regex

Perintah **grep** telah dijumpai sejak tahun 1970-an di Unix. Sejak saat itu telah berkembang beberapa *flavor* atau varian grep, masing-masing mendukung set karakter meta yang sedikit berbeda. Perintah yang bernama grep di satu Unix belum tentu memiliki kelakuan yang sama persis atau mendukung semua fitur yang ada pada varian Unix lain.

Karena banyaknya perbedaan ini, maka POSIX sebagai standar keluarga sistem operasi Unix atau yang mirip Unix, mendefinisikan standar regex. Yaitu regex dasar (basic) dan regex advanced. Salah satu fitur regex POSIX adalah dukungan terhadap *locale* atau karakter-karakter spesifik bahasa bukan Inggris. Bahasa Indonesia—sama seperti bahasa Inggris—hanya menggunakan 26 karakter Latin, jadi tidak terlalu relevan dalam hal fitur ini.

Di luar standar POSIX, bahasa-bahasa lain pun mengembangkan sintaks regex-nya sendiri. Terutama Perl. Perl adalah sebuah bahasa pemrograman yang mula-mula

berkembang di Unix dan mengambil sintaks dasarnya dari C, **awk** dan **sed**. Dua yang terakhir adalah juga perintah Unix yang banyak memanfaatkan regex. Pengembang Perl lalu banyak memperkenalkan sintaks baru sehingga regex Perl menjadi salah satu varian bahasa regex yang terampuh. Regex Perl pun menjadi popular dan banyak bahasa serta program lain yang mengikuti varian Perl ketimbang POSIX.

Saat ini bisa dikatakan flavor POSIX dan Perl merupakan 2 varian utama regex yang ada, namun sebagian besar bahasa dan tool modern mendukung sebagian besar sintaks regex ala Perl. Buku ini pun memilih membahas dengan berpatokan pada varian Perl karena kayanya fitur yang tersedia. Rata-rata bahasa mulai dari Python, Ruby, PHP hingga Java dan .NET mendukung hampir semua sintaks regex Perl. Meskipun tentunya ada sintaks yang tidak didukung dan sebaliknya ada sintaks baru yang tidak ada pada regex Perl. Bab 3 akan menjelaskan perbedaan regex di tiap-tiap bahasa. Meskipun detail dukungan regex tiap bahasa agak berbeda satu sama lain, banyak kemampuan dasar yang sama di tiap bahasa. Jadi tidak perlu kuatir karena pengetahuan regex Anda akan cukup portabel dari satu lingkungan ke lingkungan lain.

Bagaimana Menggunakan Buku Ini?

Jika Anda pemula yang sama sekali asing dengan regex, silakan baca Bab 2 dan Bab 3 dulu. Bab 2 memperkenalkan karakter-karakter meta yang membentuk regex disertai contoh-contoh. Bab 3 akan mensurvei seperti apa regex dipakai di tiap bahasa pemrograman, termasuk perbedaan atau fitur spesifik yang ada di tiap-tiap bahasa tersebut.

Bagi programer Perl atau PHP dan sebagainya yang sudah mengenal dasar-dasar regex termasuk bagaimana menggunakannya dalam bahasa masing-masing, Bab 2 dan 3 mungkin sedikit terlalu bertele-tele dan membosankan. Anda dapat langsung melihat contoh-contoh regex beserta kode program aplikasinya dalam menyelesaikan tugas-tugas nyata sehari-hari di Bab 4. Bab 4 adalah inti dari buku ini, berisi resep-resep untuk melakukan berbagai macam tugas pencocokan pola dan pengolahan teks.