CIS 18A
Introduction to Linux / Unix

Regular Expression

De Anza College
Instructor: Clare Nguyen

Topics

- Regular expression and metacharacters
- Single characters
- Anchors
- Repetition
- Operators
- Escaping metacharacters

Regular Expression

- A regular expression (abbreviated *regex*) is:
  - a series of characters
  - that you give to certain filters or utilities
  - to describe what you want to find in a line of input
- In a regular expression:
  - most characters have their *literal* meaning ('a' means you're looking for the letter 'a', space means you're looking for a space character).
  - some characters have special meaning (similar to the shell metacharacters). These are also called *metacharacters.*
- Only certain utilities (most are filters) will work with regular expressions.
- These utilities use the regular expression that you give to find a match in the input lines.

Regular Expression Engine

- To determine whether an input line has a match of a given regex, the utilities rely on the regular expression engine.
- How the regular expression engine works:
  - Walk the line of input, starting from the beginning of the line.
  - Look for a character by character match of the series of characters in the regex.
  - If there is no match by the end of the line, the regex engine concludes that the line is not a match.
  - As soon as there is a match somewhere in the line, the regex engine stops and concludes that the line is a match.
- Examples: input line: cis18a is a linux class
  a. regex: is the line is a match: cis18a is a linux class
  b. regex: as the line is a match: cis18a is a linux class
  c. regex: Linux the line is not a match
  d. regex: classes the line is not a match

Text Strings

- A text string with letters, numbers, punctuations and spaces is the simplest regular expression. Each character in the string takes on its literal meaning.
- When you use a simple text string as a regular expression, you look for an *exact* match of the text string, anywhere in the line
- Examples:
  egrep 'cis18a' input_file
  Lines that have cis18a or mycis18a or cis18abc will be sent out to screen.
  Lines with: Cis18a or CIS18A or cis 18a or cis18 will not match.
  egrep 'cis 18a ' input_file
  All lines that have cis, followed by a space, followed by 18a, followed by a space anywhere in the line will be sent out to screen.
  All other lines without the exact sequence of characters will not be sent out to screen.

Metacharacters

- In addition to simple text strings, regular expressions usually have metacharacters.
- These metacharacters make the regular expression more flexible (and thus more useful than text strings) because they allow for a range of possibilities in the search pattern. You can specify a match of 'linux' *or* 'Linux', for example.
- In the following slides are the metacharacters in the *extended* set of regular expression. The extended set contains more metacharacters in it than the original set of regular expression.
- Any character that is not a metacharacter is literal character.
- For this class we use the utility egrep to work with the extended set of regular expression.

Metacharacters that match a single character

- [characters] any one character within the [ ] is a match.
- [^characters] any one character not within the [ ] is a match .
- . any one character that is not the end-of-line character is a match (end-of-line character is the character created by hitting the enter key).

- Examples:

egrep '[Ll]inux' inputFile
Any line with Linux or linux will match.

egrep 'file[^123]' inputFile
Any line with file and *not* followed by a 1 or 2 or 3 will match.
The following will *not* match: file1 or file2.3 or file380

egrep 'lab[12][345]' inputFile
Any line with lab13 or lab14 or lab15 or lab23 or lab24 or lab25 will match.

egrep 'CIS..A' inputFile
Any line with CIS, followed by any 2 characters, followed by A will match.
This can be CIS18A or CIS35A or CISbcA or even CIS A (2 spaces between S and A), but not CISA or CIS123A

Metacharacters that are anchors

- ^ marks the beginning of the line.
  This character dictates the *position* of the *next* character in the regex.
  If used, ^ has to be the first character in the regex.
- $ marks the end of the line.
  This character matches the *position* of the *previous* character in the regex.
  If used, $ has to be the last character in the regex.

- Examples

egrep 'a' inputFile
Any line with a anywhere in the line will match.

egrep '^a' inputFile
Any line with a at the beginning of the line will match.

egrep 'a$' inputFile
Any line with a at the end of the line will match.

Metacharacters used for repetition (1 of 2)

- When a character in a regex needs to appear multiple times in a row in the match, it is easier to use the repeat metacharacters, rather than typing in a character multiple times.
- {n} n is a number, the *previous character* must appear n times.
  {n,m} n and m are numbers, the *previous character* must appear a minimum of n times and a maximum of m times.
  ? The *previous character* can appear 0 or 1 time.
  + the *previous character* must appear at least 1 time.
  * the *previous character* can appear 0 or more time.
- The metacharacters {n,m}, + and * use *greedy matching*. This means the search engine will match as many characters in the input line as possible.

  For example, if the input line is: baaaaabc
  - the regex 'a+' will match all 5 a's
  - the regex 'a{2,6}' will match all 5 a's

Metacharacters used for repetition (2 of 2)

- The * metacharacter can be counter-intuitive with what it matches. For example, in the previous example line: baaaaabc

  - the regex 'a*' will match the letter b at the beginning of the line. This is because 'a*' means 0 a or as many a's as possible, and the first b happens to match 0 a.

  - the regex 'ba*' will match the first b and then all 5 a's.

- More examples:

egrep 'ab{5,9}c' inputFile
Any line with a, followed by 5 up to 9 b's, followed by c will match.

egrep '^ *[0-9]' inputFile
Any line that starts with or without spaces in front, followed by at least 1 digit will match.

egrep '^[0-9]+$' inputFile
Any line that has only digits (and no other type of characters) will match

Notice that using the anchors ^ and $ means we're describing the whole line of text, from beginning to end. And in this case, between the beginning and the end of the line, we only allow 1 or more digits.

Metacharacters that are operators

- | means or.
  The regex 'abc|ABC' means either abc or ABC can be a match.
- ( ) means grouping
  Useful with the repetition metacharacters.
  Since the repetition metacharacters will repeat only the previous single character, if you need to repeat a group of previous characters, you need to use the ( ).

- Examples

egrep 'linux|LINUX' inputFile
Any line with linux or LINUX will match.

egrep 'abc{3}' inputFile
Any line with a, followed by b, followed by 3 c's will match.

egrep '(abc){3}' inputFile
Any line with abcabcabc (3 abc's in a row) will match.

Metacharacters used for literal meaning

- When the search engine sees a metacharacter, it uses the special meaning of the character.
- If you want to use the metacharacter for its literal meaning, you need to *escape* from the meta meaning.
- 2 ways for metacharacters to take their literal meaning:
  \ take the literal meaning of next character
  [characters] characters inside [ ] have their literal meaning

- Examples:

egrep '2.5' inputFile
Match any line with 2, followed by any single character, followed by 5
Matching lines can have: 2a5 or 2 5 or 2.5 or 215

egrep '2\.5' inputFile or egrep '2[.]5' inputFile
Match any line with 2.5

Useful Tips for Regular Expression (1 of 2)

1. For a regular expression to be flexible (and therefore more useful), it most likely will include both literal characters and metacharacters.

2. Make your regular expression as simple (as few characters) as you can.
   Examples of simple thinking:
   - 'a+' and 'a' both describe at least 1 a. Use 'a'
   - 'a{1}' and 'a' both describe 1 a. Use 'a'
   - 'aaaaaaaaa' and 'a{10}' both describe 10 a's. Use 'a{10}'
   - '^.*$' and '.*' both match everything in the line. Use '.*'
   - 'linux|Linux' and '[lL]inux' both match linux or Linux. Use '[lL]inux'
   - '^A' and '^A.*$' both describe a line that starts with A. Use '^A'

Useful Tips for Regular Expression (2 of 2)

3. Pay attention to what the repetition metacharacters will match
   Examples of non-intuitive match of repetition:
   - 'a*' will match aaaaaaaa (the obvious case), but it also will match bcd (the not so obvious case).
   - '^a+$' means that the line has to have at least 1 a, but
   - '^a*$' means the line can be empty (no character).

4. Don't forget the anchors ^ and $ when you need to describe the entire line. This typically happens when you're looking for:
   - *exactly* n numbers of a's and nothing else: '^a{n}$'
   - *only* a's and nothing else: '^a+$'
   - no a's: '^[^a]+$'

   If the 3 regex above don't have both anchors, then the text string: aaaabc will match all 3 of them

The End

Congratulations. You've reached the end of the material for CIS18A