Regular Expressions
Pattern Expansion
Unix Commands
Job submission

Regular Expressions

- A regular expression is a pattern which matches some regular (predictable) text.
- Regular expressions are used in many Unix utilities.
  - like grep, sed, vi, emacs, awk, ...
- The form of a regular expression:
  - It can be plain text ...

    grep unix file (matches all the appearances of unix)
  - It can also be special text ...
  - grep '[uU]nix' file (matches unix and Unix)

Regular Expressions and File Wildcarding

- Regular expressions are different from file name wildcards.
  - Regular expressions are interpreted and matched by special utilities (such as grep).
  - File name wildcards are interpreted and matched by shells.
  - They have different wildcarding systems.
  - File wildcarding takes place first!

grep ‘[uU]nix’ file

grep [uU]nix file

Regular Expression Wildcards

- A dot . matches any single character

  a.b matches axb, a$b, abb, a.b

  but does not match ab, axxb, a$bccb

- * matches zero or more occurrences of the previous single character pattern

  a*b matches b, ab, aab, aaab, aaaab, …

  but doesn't match axb

- What does the following match?

  .*

Character Ranges

- Matching a set or range of characters is done with [...]
  - [wxyz] - match any of wxyz
    [u-z] - match a character in range u - z
- Combine this with * to match repeated sets
  - Example: [aeiou]* - match any number of vowels
- Wildcards lose their specialness inside [...]
  - If the first character inside the [...] is], it loses its specialness as well
  - Example: '[])}]' matches any of those closing brackets

Match Parts of a Line

- Match beginning of line with ^ (caret)

  ^TITLE

  - matches any line containing TITLE at the beginning
  - ^ is only special if it is at the beginning of a regular expression
- Match the end of a line with a $ (dollar sign)

  FINI$

  - matches any line ending in the phrase FINI
  - $ is only special at the end of a regular expression
  - Don't use $ and double quotes (problems with shell)
- What does the following match? ^WHOLE$

Matching Parts of Words

- Regular expressions have a concept of a “word” which is a little different than an English word.
  - A word is a pattern containing only letters, digits, and underscores (_)
- Match beginning of a word with **\<**
  - \<Fo matches Fo if it appears at the beginning of a word
- Match the end of a word with \>
  - ox\> matches ox if it appears at the end of a word
- Whole words can be matched too: \<Fox\>

More Regular Expressions

- Matching the complement of a set by using the ^
  - [^aeiou] - matches any non-vowel
  - ^[^a-z]*$ - matches any line containing no lower case letters
- Regular expression escapes
  - Use the \ (backslash) to “escape” the special meaning of wildcards
    - CA*Net
    - This is a full sentence\.
    - array\[3]
    - C:\\DOS
    - \[.*\]

Regular Expressions Recall

- A way to refer to the most recent match
- To remember portions of regular expressions
  - Surround them with \(...\)
  - Recall the remembered portion with \n where n is 1-9
    - Example: '^\([a-z]\)\1'
      - matches lines beginning with a pair of duplicate (identical) letters
    - Example: '^.*\([a-z]*\).*\1.*\1'
      - matches lines containing at least three copies of something which consists of lower case letters

Matching Specific Numbers of Repeats

- X\{m,n\} matches m -- n repeats of the one character regular expression X
  - E.g. [a-z]\{2,10\} matches all sequences of 2 to 10 lower case letters
- X\{m\} matches exactly m repeats of the one character regular expression X
  - E.g. #\{23\} matches 23 #s
- X\{m,\} matches at least m repeats of the one character regular expression X
  - E.g. ^[aeiou]\{2,\} matches at least 2 vowels in a row at the beginning of a line
- .\{1,\} matches more than 0 characters

Pattern Expansion in bash

Substring removal

- ${PARAMETER#PATTERN}
- ${PARAMETER##PATTERN}
- ${PARAMETER%PATTERN}
- ${PARAMETER%%PATTERN}

This one can **expand only a part** of a parameter's value, **given a pattern to describe what to remove** from the string.

Pattern Expansion in bash

Example string (*just a quote from a big man*):

- MYSTRING="Be liberal in what you accept, and conservative in what you send"

From the beginning:

${PARAMETER#PATTERN} and

${PARAMETER##PATTERN}

Pattern Expansion in bash

This form is to remove the described pattern trying to **match it from the beginning of the string**. The operator "#" will try to remove the shortest text matching the pattern, while "##" tries to do it with the longest text matching.

${MYSTRING#* } => liberal in what you accept, and conservative in what you send

${MYSTRING##* }=> send

Pattern Expansion in bash

From the end

${PARAMETER%PATTERN} and

${PARAMETER%%PATTERN}

In the second form everything will be the same, except that Bash now tries to match the pattern from the end of the string:

${MYSTRING% *} => Be liberal in what you accept, and conservative in what you

${MYSTRING%% *} => Be

Pattern Expansion in bash

Common use

- ***How the heck does that help to make my life easier?***

Well, maybe the most common use for it is to **extract parts of a filename**. Just look at the following list with examples:

- **Get name without extension**
  - ${FILENAME%.*}
  - ⇒ bash_hackers.txt
- **Get extension**
  - ${FILENAME##*.}
  - ⇒ bash_hackers.txt

Pattern Expansion in bash

- **Get directory name**
  - ${PATHNAME%/*}
  - ⇒ /home/bash/bash_hackers.txt
- **Get filename**
  - ${PATHNAME##*/}
  - ⇒ /home/bash/bash_hackers.txt

These are the syntaxes for filenames with a single extension. Depending on your needs, you might need to adjust shortest/longest match.

Best Additional Unix Commands

- history
- uptime
- top
- ps
- tar
- diff
- sort
- gzip/bzip2
- crontab
- chown
- chgrp
- umask
- tail
- head
- ping
- uptime
- which
- wget
- kill
- nice