Lanka Education and Research Network

LEARN

29th Nov 2022 : Session 01

Deepthi Gunasekara / LEARN

What is a filter in linux?

- Programs that
  - Take plain text(either stored in a file or produced by another program) as standard input
    - Transforms it into a meaningful format
      - Then returns it as standard output.

Linux has a number of filters and let's see few:

- Grep
- Find
- SED
- AWK

What is grep?

- Grep : “global regular expression print
- Command used in
  - ✓ searching and matching text files contained in the regular expressions(The text search pattern).
- Linux / Unix command-line tool used to search for a string of characters in a specified file
- Pre-installed in every Linux distribution.
- Handy when searching through large log files

How do I grep files in Linux?

- Type grep
  - then the pattern we're searching for
    - Finally the name of the file (or files) we're searching in.

Grep….Cont…

Syntax:
grep [options] [pattern] [file]

Examples:

- Match all lines that start with ‘hello’.
  $grep “^hello” file_hello

- Match all lines that contain the word hello in upper-case or lower-case
  $ grep -i “hello” ” file_hello

- Checking for the given string in multiple files(Files names are started with demo).
  $ grep "this" demo_*

What is find in linux?

- Command-line utility that
  - ✓ Used to search and locate the list of files and directories based on conditions specify for files that match the arguments.
  - ✓ if another action is requested, performs that action on each matched object

- Used in a variety of conditions like find files by :
  - ✓ permissions
  - ✓ Users
  - ✓ Groups
  - ✓ file types
  - ✓ Date
  - ✓ Size
  - ✓ other possible criteria

find…Cont…

Syntax:
find [options] [path...] [expression]

Examples:

- Find all the files under /home directory with the name hello.txt.
  # find /home -name hello.txt

- Find all the files whose name is hello.txt and contains both capital and small letters in /home directory.
  # find /home –iname hello.txt

- Find all php files in a directory.
  # find . -type f -name "*.php"

What is sed?

- Linux stands for ”Stream EDitor”
- Used on Unix systems to edit files quickly and efficiently
- Operations like
  - ✓ selecting the text
  - ✓ substituting text
  - ✓ modifying an original file
  - ✓ adding lines to text
  - ✓ deleting lines from the text

Sed…Cont…

Syntax:
sed options [script][inputfile]

Examples:

- To output the content between lines 3 and 5 of the file hello.txt:
  $sed -n '3,5p' hello.txt

- To replace the second occurrence of the word with sed, pass a number to the g argument
  $sed s/amazing/super/g2 hello.txt

- substitute words inside a specific range.
  $sed '2,5s/amazing/super/g' hello.txt

What awk?

- Abbreviated from the names of the developers – Aho, Weinberger, and Kernighan.
- Linux tool and programming language that **allows users to process and manipulate data and produce formatted reports**
- Scripting language used for manipulating data and generating reports.
- no compiling and allows the user to use variables, numeric functions, string functions, and logical operators.

Awk….cont….

- Utility, enables a programmer to write tiny but effective programs in the form of statements
  - ✓ text patterns that are to be searched for in each line of a document and the action that is to be taken when a match is found within a line.

- Mostly used for pattern scanning and processing.
  - ✓ It searches one or more files to see if they contain lines that matches with the specified patterns and then perform the associated actions

Awk cont….

Syntax:
awk options 'selection _criteria {action }' input-file > output-file

Examples:

- Print the lines which match the given pattern.
  $ awk '/manager/ {print}' employee.txt

- To return the second column/item from employee.txt
  $ awk '{print $2}' employee.txt

- Printing lines with more than 10 characters
  $ awk 'length($0) > 10' employee.txt

REGEX…..

- Regular expression is also called **regex or regexp**.
- A very powerful tool in Linux
- A pattern for a matching string that follows some pattern.
- Can be used in a variety of programs like grep, sed, vi and many more

Many languages allow programmers to define regexes and then use them to:

- **Validate** that a piece of text (or a portion of that text) matches some pattern
- **Find** fragments of some text that match some pattern
- **Extract** fragments of some text
- **Replace** fragments of text with other text

REGEX…..

There are three versions of regular expressions syntax:

- Basic Regular Expressions
- Interval Regular Expressions
- Extended Regular Expressions

Depending on tool or programs, one or more of these versions can be used.

REGEX…Cont....

Basic Regular Expressions

Some of the commonly used commands with Regular expressions are tr, sed, vi and grep. Listed below are some of the basic Regex.

Symbol	Descriptions
.	replaces any character
^	matches start of string
$	matches end of string
*	matches up zero or more times the preceding character
\	Represent special characters
()	Groups regular expressions
?	Matches up exactly one character

REGEX…Cont....

Interval Regular expressions

These expressions tell us about the number of occurrences of a character in a string. They are

Expression	Description
{n}	Matches the preceding character appearing 'n' times exactly
{n,m}	Matches the preceding character appearing 'n' times but not more than m
{n, }	Matches the preceding character only when it appears 'n' times or more

REGEX…Cont....

Extended regular expressions

These regular expressions contain combinations of more than one expression. Some of them are:

Expression	Description
\+	Matches one or more occurrence of the previous character
\?	Matches zero or one occurrence of the previous character

REGEX….Cont….

Regex Syntax Summary

➢ **Character**:

- All characters, except those having special meaning in regex, matches themselves.

Ex: The regex x matches substring "x"; regex 9 matches "9"; regex =matches "="; and regex @ matches "@".

➢ **Special Regex Characters**:

- These characters have special meaning in regex (to be discussed below): ., +, *, ?, ^, $, (,), [,], {, }, |, \.

REGEX….Cont….

- **Escape Sequences (\char)**:
  - To match a character having special meaning in regex, you need to use a escape sequence prefix with a backslash (\). E.g., \. matches "."; regex \+ matches "+"; and regex \(matches "(".
  - You also need to use regex \\ to match "\" (back-slash).
  - Regex recognizes common escape sequences such as \n for newline, \t for tab, \r for carriage-return, \nnn for a up to 3-digit octal number, \xhh for a two-digit hex code, \uhhhh for a 4-digit Unicode, \uhhhhhhhh for a 8-digit Unicode.

REGEX….Cont….

- **A Sequence of Characters (or String)**:

  - Strings can be matched via combining a sequence of characters (called sub-expressions).

    E.g., the regex Saturdaymatches "Saturday". The matching, by default, is case-sensitive, but can be set to case-insensitive via *modifier*.

- **OR Operator (|)**:

  E.g., the regex four|4 accepts strings "four" or "4".

REGEX….Cont….

- **Character class (or Bracket List)**:
  - [...]: Accept ANY ONE of the character within the square bracket,

  e.g., [aeiou] matches "a", "e", "i", "o" or "u".

  - [.-.] (Range Expression): Accept ANY ONE of the character in the *range*, e.g., [0-9] matches any digit; [A-Za-z] matches any uppercase or lowercase letters.

  - [^...]: NOT ONE of the character,

  e.g., [^0-9] matches any non-digit.

  - Only these four characters require escape sequence inside the bracket list: ^, -,], \.

REGEX….Cont….

- **Occurrence Indicators (or Repetition Operators)**:
  - +: one or more (1+),

  e.g., [0-9]+ matches one or more digits such as '123', '000'.

  - *: zero or more (0+),

  e.g., [0-9]* matches zero or more digits. It accepts all those in [0-9]+ plus the empty string.

  - ?: zero or one (optional),

  e.g., [+-]? matches an optional "+", "-", or an empty string.

  - {m,n}: m to n (both inclusive)
  - {m}: exactly m times
  - {m,}: m or more (m+)

REGEX….Cont….

- **Metacharacters**: matches a character
  - . (dot): ANY ONE character except newline. Same as [^\n]
  - \d, \D: ANY ONE digit/non-digit character. Digits are [0-9]
  - \w, \W: ANY ONE word/non-word character. For ASCII, word characters are [a-zA-Z0-9_]
  - \s, \S: ANY ONE space/non-space character. For ASCII, whitespace characters are [\n\r\t\f]

REGEX….Cont….

➢ **Position Anchors**:

- Does not match character, but position such as start-of-line, end-of-line, start-of-word and end-of-word.
- ^, $: start-of-line and end-of-line respectively.
  E.g., ^[0-9]$ matches a numeric string.

- \b: boundary of word, i.e., start-of-word or end-of-word. E.g., \bcat\b matches the word "cat" in the input string.

- \B: Inverse of \b, i.e., non-start-of-word or non-end-of-word.
- \<, \>: start-of-word and end-of-word respectively, similar to \b. E.g., \<cat\> matches the word "cat" in the input string.

- \A, \Z: start-of-input and end-of-input respectively.

REGEX….Cont….

Example: Numbers [0-9]+ or \d+

- The [...], known as *character class* (or *bracket list*), encloses a list of characters. It matches any SINGLE character in the list. In this example, [0-9] matches any SINGLE character between 0 and 9 (i.e., a digit), where dash (-) denotes the *range*.

- The +, known as *occurrence indicator* (or *repetition operator*), indicates one or more occurrences (1+) of the previous sub-expression. In this case, [0-9]+ matches one or more digits.

- A regex may match a portion of the input (i.e., substring) or the entire input. In fact, it could match zero or more substrings of the input (with global modifier).

REGEX….Cont….

Example: Numbers [0-9]+ or \d+

- The [...], known as *character class* (or *bracket list*), encloses a list of characters. It matches any SINGLE character in the list. In this example, [0-9] matches any SINGLE character between 0 and 9 (i.e., a digit), where dash (-) denotes the *range*.

- The +, known as *occurrence indicator* (or *repetition operator*), indicates one or more occurrences (1+) of the previous sub-expression. In this case, [0-9]+ matches one or more digits.

- A regex may match a portion of the input (i.e., substring) or the entire input. In fact, it could match zero or more substrings of the input (with global modifier).

REGEX….Cont….

Example: Numbers [0-9]+ or \d+

- This regex matches any numeric substring (of digits 0 to 9) of the input. For examples,

  1. If the input is "abc123xyz", it matches substring "123".
  2. If the input is "abcxyz", it matches nothing.
  3. If the input is "abc00123xyz456_0", it matches substrings "00123", "456" and "0" (three matches).

- Take note that this regex matches number with leading zeros, such as "000", "0123" and "0001", which may not be desirable.

REGEX….Cont….

Example: Numbers [0-9]+ or \d+

- You can also write \d+, where \d is known as a metacharacter that matches any digit (same as [0-9])

There are more than one ways to write a regex!

Take note : many programming languages (C, Java, JavaScript, Python) use backslash \ as the prefix for escape sequences (e.g., \n for newline), and you need to write "\\d+" instead.

Again…….

- Grep, sed, find and AWK are all **standard Linux tools that are able to process text**.
  - ✓ Each of these tools can read text files line-by-line and use regular expressions to perform operations on specific parts of the file


THANK
YOU