ULI101: INTRODUCTION TO UNIX / LINUX AND THE INTERNET

WEEK 9 LESSON I

REGULAR EXPRESSIONS VS FILENAME EXPANSION / SIMPLE AND COMPLEX REGULAR EXPRESSIONS

PHOTOS AND ICONS USED IN THIS SLIDE SHOW ARE LICENSED UNDER CC BY-SA

LESSON I TOPICS

Regular Expressions

- Definition / Purpose
- Regular Expressions vs. Filename Expansion

Simple and Complex Regular Expressions

- Simple (Literal) Regular expressions using grep
- Complex Regular Expressions using grep
- Demonstration

Perform Week 9 Tutorial

- Investigation I
- Review Questions (<u>Simple</u> and <u>Complex</u> Regular Expressions Parts A and B)

Definition

A **regular expression** ... is a sequence of characters that define a **search pattern**. Usually, such patterns are used by string searching algorithms for "**find**" or "**find and replace**" operations on strings, or for **input validation**.

Reference: https://en.wikipedia.org/wiki/Regular_expression



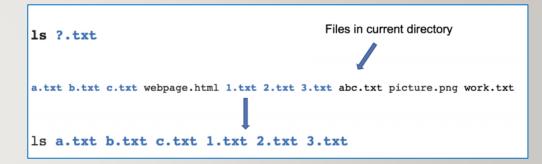


Regular Expressions vs Filename Expansion

In a previous lesson, you learned **filename expansion** symbols that allow the Linux shell to **expand** filenames as **arguments** (referred to as "globbing").

This is used for command file management and file manipulation commands including:

```
ls, rm, mv, cp, cat, less, more, head,
tail, sort, uniq, grep, tr, cut and wc
```





Regular Expressions vs Filename Expansion

Regular expressions are used to search, edit and manipulate <u>text</u>. This can represent text <u>contained</u> in a file <u>or</u> within a pipeline command.

Regular expressions are used with commands that match patterns contained in text such as: grep, egrep, man, more, less, vi, sed and awk

Simple (literal) Regular Expressions With Linux Commands



A simple regular expression is a collection of **characters** (for example words or phases).

Although we will later discuss several Linux commands that use regular expressions, the **grep** Linux command is useful to **learn** to display lines of text that **match** a regular expression.

Example:

grep Linux document.txt

cat document.txt

I like Linux
It is different than Windows
I find Linux useful

grep Linux document.txt

I like Linux I find Linux useful

Regular Expressions With Linux Pipeline Commands

Regular expressions can also be used to manipulate text within Linux Pipeline Commands.



The **grep** command can act as a **filter** to match text patterns. In turn, the **stdout** from that filter can be further processed by other *filters* throughout the *Linux pipeline command*.



Examples:

```
ls | grep txt
who | grep khan | head -20
```

```
ls
1.txt 3.bash a.doc a.txt b.docx b.txt
ls | grep txt
1.txt
a.txt
b.txt
```

Instructor Demonstration

Your instructor will demonstrate examples of using simple regular expressions with the grep command.



More Precise Pattern Matching

The problem with using simple (literal) regular expressions is that only **simple** or **general** patterns are matched.

For example, the **pattern:** the would match larger words such as there, they, either, them, their, in addition to the word the.

There are also other types of patterns you may want to search such as **location** of pattern at the beginning or ending of a string, **number** of characters (or character classes) or the **number of occurrences** of a *character* or *pattern*.

We can use **complex** and **extended** regular expressions for more precise matches. We will discuss **complex** regular expressions in this lesson.



Complex Regular Expressions Symbols

Complex Regular Expressions use **symbols** to help match text for more **precise** or **complex** patterns.

The most common complex regular expression symbols are displayed below:

```
Anchors ^, $
Characters .
Character Class [ ], [^ ]
Zero or More Occurrence *
```



Complex Regular Expressions Symbols

Anchors: ^, \$

Anchors are used to "anchor" the match at a **specific** position (at beginning or ending of a string of text).

The \$\symbol anchors the pattern at the **beginning** of the string. The \$\symbol anchors the pattern at the **end** of the string.

Examples:

```
grep "^Beginning" data.txt
grep "end$" data.txt
```

cat data.txt

Beginning of the line
This is not at the beginning
This is at the end
Beginning of line and the end
Not at beginning and end not so

grep "^Beginning" data.txt

Beginning of the line Beginning of line and the end

grep "end\$" data.txt

This is at the end Beginning of line and the end

Complex Regular Expressions Symbols

Single Character: .

The period symbol "." is used to represent a **single character** which could represent **any** character.

This symbol (or sequence of period symbols) are effective when used with **anchors**.

Examples:

```
grep "^.$" data.txt
grep "^....$" data.txt
```

```
cat data.txt
Hello
Therefore
Hi
I
isn't

grep "^.$" data.txt
I

grep "^.....$" data.txt
Hello
isn't
```

Complex Regular Expressions Symbols

Character Class: [], [^]

Works like the Single Character symbol, but with **restrictions**.

The ^ symbol with the character class means **opposite** of the contents within the character class.

This symbol (or sequence of these symbols) are effective when used with **anchors**.

Examples:

```
grep "^[a-z][a-z] [a-z] data.txt
grep "[^a-zA-Z]$" data.txt
```

```
cat data.txt
abc123
12abcdef
abc.
XYZ
123abc+

grep "^[a-z][a-z][a-z]" data.txt
abc123
abc.

grep "[^a-zA-Z]$" data.txt
abc123
abc.
123abc+
```

Complex Regular Expressions Symbols

Zero or More Occurrence(s) *

This symbol means zero of more occurrences of the previous character.

People learning about regular expressions get **confused** with this symbol thinking that it means zero or any character, but that would require the use of two symbols: . *

Examples:

```
grep "Linux i*" data.txt
grep "I*s an" data.txt
grep "^[0-9].*[0-9]$" myfile.txt
```

data.txt

Linux is an OS
Linux iis and OS
Linux is a choice
is true
iis true
iiis true
true

grep "Linux i*" data.txt

Linux is an OS Linux iis and OS Linux is a choice

grep "i*s an" data.txt

Linux is an OS Linux iis and OS

Instructor Demonstration

Your instructor will demonstrate examples of using complex regular expressions with the grep command.



Tip: Creating a Reference Sheet

It is a good idea to keep symbols for Filename Expansion and Regular Expressions **separate** since there is some overlapping similar symbols that have different purposes.

It is recommended to write-out these separate set of symbols on a **sheet of paper** for reference.

REGULAR EXPRESSION SYMBOLS

- ^ anchor at beginning
- \$ anchor at ending
- . single character (any character)
- [], [^] single character (restrictions)
- * zero or more occurrences of preceding character

HOMEWORK

Getting Practice

Perform Week 9 Tutorial:

(Due: Friday Week 10 @ midnight for a 2% grade):

- INVESTIGATION I: SIMPLE & COMPLEX REGULAR EXPRESSIONS
- LINUX PRACTICE QUESTIONS

Review Questions (Simple and Complex Regular Expressions Parts A and B)