OSL640: INTRODUCTION TO OPEN SOURCE SYSTEMS

WEEK 9 LESSON 1

REGULAR EXPRESSIONS VS FILENAME EXPANSION / SIMPLE AND COMPLEX REGULAR EXPRESSIONS

LESSON 1 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 1
- Review Questions (**Simple** and **Complex** **Regular Expressions** Parts **A** and **B**)

REGULAR EXPRESSIONS

Definition

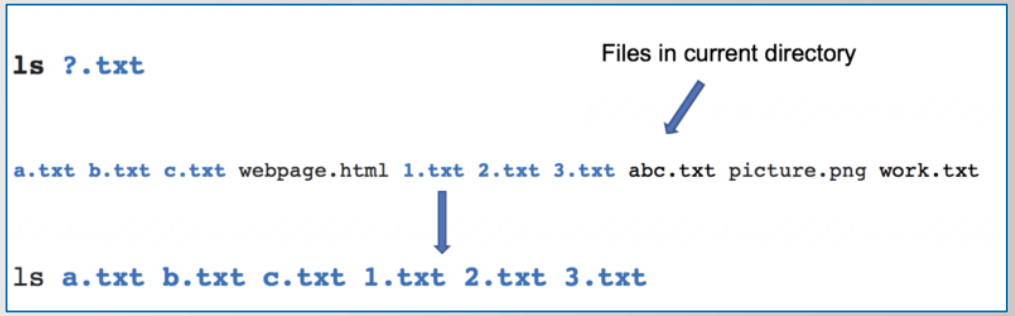
*A **regular expression** … is a sequence of characters that define a **search pattern**. Usually, such patterns are used by string searching algorithms for "**find**" or "**find and replace**" operations on strings, or for **input validation**.*

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

REGULAR EXPRESSIONS

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

Regular Expressions vs Filename Expansion

Regular expressions are used to **search**, **edit** and **manipulate** **text**.
This can represent text contained in a **file** or 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`**

REGULAR EXPRESSIONS

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

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
```

REGULAR EXPRESSIONS

Instructor Demonstration

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

REGULAR EXPRESSIONS

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.

REGULAR EXPRESSIONS

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 `*`

REGULAR EXPRESSIONS

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
```

REGULAR EXPRESSIONS

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
```

REGULAR EXPRESSIONS

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+
```

REGULAR EXPRESSIONS

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
```

REGULAR EXPRESSIONS

Instructor Demonstration

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

REGULAR EXPRESSIONS

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.

```
FILENAME EXPANSION SYMBOLS


? – single character (any character)

[ ],[! ] – single character
            ( with restrictions)

* - zero of more characters
    (any character)
```

```
REGULAR EXPRESSION SYMBOLS

^ - anchor at beginning

$ - anchor at ending

. – single character (any character)

[ ] , [^] – single character
            (restrictions)

* - zero or more occurrences of
    preceding character

.* - zero or any number of characters
     (any character)
```

REGULAR EXPRESSIONS

Getting Practice

To get practice perform **Week 9 Tutorial:**

- INVESTIGATION 1: SIMPLE & COMPLEX REGULAR EXPRESSIONS
- LINUX PRACTICE QUESTIONS

OSL640: INTRODUCTION TO OPEN SOURCE SYSTEMS

WEEK 9: LESSON 2

EXTENDED REGULAR EXPRESSIONS
LINUX COMMANDS THAT USE REGULAR EXPRESSIONS

LESSON 2 TOPICS

Extended Regular Expressions

- Definition / Purpose
- Extended Regular Expressions Symbols
- Instructor Demonstration

Other Linux Commands That Use Regular Expressions

- `man` , `more` , `less` , `vi` , `sed` , `awk`

Perform Week 9 Tutorial

- Investigation 2
- Review Questions (**Extended Regular Expressions**, Parts **A** and **B**)

EXTENDED REGULAR EXPRESSIONS

Extended Regular Expressions

Extended Regular Expressions consist of additional special characters that "**extend**" the capability of regular expressions.

We will discuss three types of **extended regular expressions**:

Repetition: `{min, max}` , `?` , `+`

Grouping: `(` `)`

`Or` **Condition:** `|`

EXTENDED REGULAR EXPRESSIONS

AAAA

Repetition

The extended regular expression symbol consists of the
minimum and/or **maximum** number of repetitions contained within braces `{ }`

Usage:

`{min,max}`

Examples:

`a{2,5}`	**2 to 5** occurrences of the character **a**
`[0-9]{1,}`	**1 or more** occurrences of a **number**
`[0-9]+`	(shortcut method)
`[a-z]{0,1}`	**zero or 1** occurrence of a **lowercase letter**
`[a-z]?`	(shortcut method)

EXTENDED REGULAR EXPRESSIONS

AAAA

Repetition Extended Regular Expression Example

If you issue the **grep** command without options with **extended** regular expressions, the command **will NOT work**.

When using the grep command with extended regular expressions you *need to use* ***egrep*** *or* ***grep -E***

Examples:

```
egrep "^[0-9]{1,}$" data.txt
egrep "^[+-]{0,1}[0-9]{1,}$" data.txt
egrep "^[0-9]{1,}[.]{0,1}[0-9]{0,}$" data.txt

grep -E  "^[0-9]{1,}$" data.txt
grep -E "^[+-]{0,1}[0-9]{1,}$" data.txt
grep -E "^[0-9]{1,}[.]{0,1}[0-9]{0,}$" data.txt
```

```
cat data.txt
123
+45
+++37
-67.89
--57.6
-78...4
12.6
+26.887

egrep "^[0-9]{1,}$" data.txt
123

egrep "^[+-]{0,1}[0-9]{1,}$" data.txt
123
+45

egrep "^[0-9]{1,}[.]{0,1}[0-9]{0,}$" data.txt
123
12.6
```

EXTENDED REGULAR EXPRESSIONS

(pattern)

Grouping

If you want to search for repetition for a **group** of **characters**, a **word**, or a **phase**, you can enclose them within brackets **( )**

Examples:

```
egrep "(the ){2,}" data.txt
egrep "(lazy fox ){2,3}" data.txt
```

```
cat data.txt
The lazy fox jumped over dog
Time to go to the the store
I like to go to the movies
I act like a lazy fox lazy fox lazy fox
Don't be a lazy fox

egrep "(the ){2,}" data.txt
Time to go to the the store

egrep "(lazy fox ){2,3}" data.txt
I act like a lazy fox lazy fox lazy fox
```

REGULAR EXPRESSIONS

(this |that)

`Or` Condition

The | symbol is used as the "or" symbol to provide **alternatives** within a **group**.

Examples:

```
egrep "(this | that ){1,}" data.txt
egrep "(a|b|c){3,}" data.txt
```

```
cat data.txt
I know this is the day
Because that is correct
We don't know that it is sunny
I know how to cccamp
I waaaaant a tissue
Can a bbborrow a cup of sugar?

egrep "(this | that ){1,}" data.txt
I know this is the day
Because that is correct
We don't know that it is sunny

egrep "(a|b|c){3,}" data.txt
I know how to cccamp
I waaaaant a tissue
Can a bbborrow a cup of sugar?
```

REGULAR EXPRESSIONS

Instructor Demonstration

Your instructor will demonstrate examples of using
Extended Regular expressions with the **egrep** command.

REGULAR EXPRESSIONS

Other Linux Commands that Use Extended Regular Expressions

There are other Linux commands / utilities in addition to ***`grep`*** or ***`egrep`***
that use regular expressions.

You have already used a few of these commands like:
`man`, **`more`**, **`less`**, and **`vi`**.

Other commands like **`sed`** and **`awk`** will be taught in a future lesson.

REGULAR EXPRESSIONS

Other Linux Commands that Use Extended Regular Expressions

`man` , `more` , `less`

When searching for patterns using the **`man`**, **`more`**, or **`less`** commands**,** you specify a regular expression with a **forward slash /**

*Example with **man ls** command:*

`/classify`

```
-F, --classify
       append indicator (one of */=>@|) to entries

--file-type
       likewise, except do not append '*'

--format=WORD
       across -x, commas -m, horizontal -x, long -l,

--full-time
       like -l --time-style=full-iso
```

REGULAR EXPRESSIONS

Other Linux Commands that Use Extended Regular Expressions

`vi`

The **vi** text editor use regular expressions to search and manipulate (edit) text within a text document.

Examples:

`/pattern` – search for pattern in text file
`:%s/osl640/OSL640/g` – search and replace text globally (all lines)

```
I am taking the course ULI101
There are a lot of commands taught in ULI101
I am over half way in the ULI101 course
I like Linux

~
~
~
~
~
```

REGULAR EXPRESSIONS

Using Regular Expressions with Linux Commands other than grep

`awk` , `sed`

The **awk** and **sed** Linux utilities are used to **manipulate** text, from files or via Linux pipeline commands.

You will learn how to use these commands in a **later** lesson.

REGULAR EXPRESSIONS

Instructor Demonstration

Your instructor will demonstrate examples of using **Extended Regular Expressions** with the **man**, **more**, **less** and **vi** utilities.

REGULAR EXPRESSIONS

Getting Practice

To get practice perform **Week 9 Tutorial:**

- INVESTIGATION 2: EXTENDED REGULAR EXPRESSIONS
- INVESTIGATION 3: OTHER COMMANDS THAT USE REGULAR EXPRESSIONS
- LINUX PRACTICE QUESTIONS