

Week Two, Lecture 2

Agenda

1. Shell Scripting
2. RegEx
3. sed, awk

Video on Shell Scripting

<https://www.youtube.com/watch?v=eiBVlxxu3so>

Regex Utility

<https://regex101.com/>

- RegEx: A **regular expression** is a special text string for describing a search pattern.
- In Windows, a wildcard notation such as *.txt finds all text files in a directory.
- The regex equivalent is `^.*\.txt$`.
- Why?
 - Programatically matching data in records to a pattern
 - Validating user data input

Meta Characters

MetaCharacter	Usage
.	Any one character
[]	Any enclosed character
*	Zero or more of the preceding character
?	Zero or one of the preceding character
+	One or more of the preceding character
^	Anchor - The beginning of the string
\$	Anchor - The end of the string
\	Escape character (treat what follows as a literal, NOT a regex command character)
	"or"
{m,n}	The preceding character occurs from m to n times
\b	Anchor – beginning of a word
[[:blank:]]	Space or Tab

RegExpr		Means	RegExpr		Means
[A-H]	➔	[ABCDEFGH]	[^AB]	➔	Any character except A or B
[A-Z]	➔	Any uppercase alphabetic	[A-Za-z]	➔	Any alphabetic
[0-9]	➔	Any digit	[^0-9]	➔	Any character except a digit
[[a]	➔	[or a	[[a]	➔] or a
[0-9\ -]	➔	digit or hyphen	[^\^]	➔	Anything except^

Sequence Operator

In a sequence operator, if a series of atoms are shown in a regular expression, there is no operator between them.

<code>dog</code>	→	matches the pattern "dog"
<code>a..b</code>	→	matches "a" , any two characters, and "b"
<code>[2-4][0-9]</code>	→	matches a number between 20 and 49
<code>[0-9][0-9]</code>	→	matches any two digits
<code>^\$</code>	→	matches a blank line
<code>^.\$</code>	→	matches a one-character line
<code>[0-9]-[0-9]</code>	→	matches two digits separated by a "-"



\\"\\(8

5{2}

[J,j]a

[J,j]a.{1,10}

[J,j]a[^"]*"\\(

\\b[M|m]\\b[F|f]

b[A-Z|a-z]+@[A-Z|a-z]+\\. [A-Z|a-z]+

[[:blank:]]

Short Form Repetition Operators: * + ?

Formats

*



special case: matches previous atom zero or more times

+



special case: matches previous atom one or more times

?



special case: matches previous atom 0 or one time only

Examples

BA*



B, BA, BAA, BAAA, BAAAA, ...

B.*



B, BA ... BZ, BAA ... BZZ,
BAAA ... BZZZ, ...

.*



zero or more characters

.*



one or more characters

[0-9]?



zero or one digit



Examples

"^The"	matches any string that starts with "The".
"of despair\$"	matches a string that ends in with "of despair".
"^abc\$"	a string that starts and ends with "abc" - effectively an exact match comparison.
"notice"	a string that has the text "notice" in it.
"ab*"	matches a string that has an a followed by zero or more b's ("ac", "abc", "abbc", etc.)
"ab+"	same, but there's at least one b ("abc", "abbc", etc., but not "ac")
"ab?"	there might be a single b or not ("ac", "abc" but not "abbc").
"a?b+\$"	a possible 'a' followed by one or more 'b's at the end of the string: Matches any string ending with "ab", "abb", "abbb" etc. or "b", "bb" etc. but not "aab", "aabb" etc.

Examples

"ab{2}"	matches a string that has an a followed by exactly two b's ("abb")
"ab{2,}"	a followed by at least two b's ("abb", "abbbb", etc.)
"ab{3,5}"	a followed by from three to five b's ("abbb", "abbbb", or "abbbbb")
"a(bc)*"	matches a string that has an a followed by zero or more copies of the sequence "bc"
"a(bc){1,5}"	a followed by one through five copies of "bc"
"hi hello"	matches a string that has either "hi" or "hello" in it
"(b cd)ef"	a string that has either "bef" or "cdef"
"(a b)*c"	a string that has a sequence of alternating a's and b's ending in a c
"a.[0-9]"	matches a string that has an a followed by one character and a digit
"^{3}\$"	a string with exactly 3 characters
"[ab]"	matches a string that has either an a or a b (that's the same as "a b")
"[a-d]"	a string that has lowercase letters 'a' through 'd' (that's equal to "a b c d" and even "[abcd]")
"^[a-zA-Z]"	a string that starts with a letter
"[0-9]%"	a string that has a single digit before a percent sign
",[a-zA-Z0-9]\$"	a string that ends in a comma followed by an alphanumeric character

Further Reading:

<https://regex101.com/>

<http://regexone.com/>

<http://www.zytrax.com/tech/web/regex.htm>

<http://docs.python.org/2/howto/regex.html>

AWK

- a programming language designed for text processing
- Used for processing regular expressions in a script
- Used when the text is in file / delimited field format
- typically used as a data extraction and reporting tool
- a powerful standard feature of most Unix-like operating systems.

awk operations:

- scans a file line by line
- splits each input line into fields
- compares input line/fields to pattern
- performs action(s) on matched lines

Useful for:

- transforming data files
- Producing formatted reports

Programming constructs:

- format output lines for reports
- arithmetic and string operations
- conditionals and loops

Basic awk Script

- consists of patterns & actions:
`pattern {action}`
 - if pattern is missing, action is applied to all lines
 - if action is missing, the matched line is printed
 - must have either pattern or action

Example:

```
awk '/for/' testfile
```

- prints all lines containing string “for” in testfile



Basic Terminology: input file

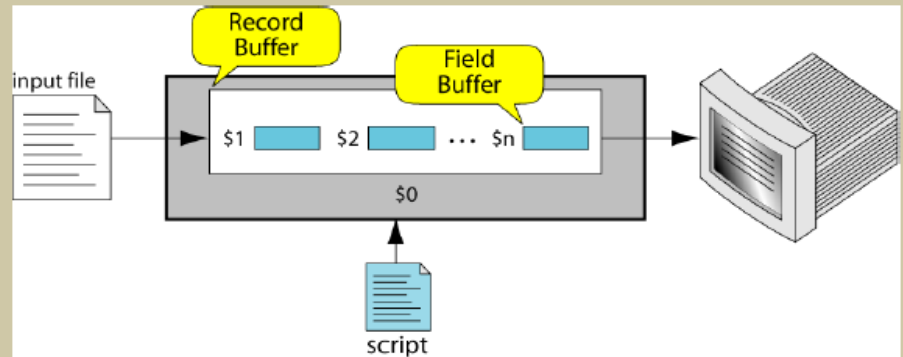
- A field is a unit of data in a line
- Each field is separated from the other fields by the field separator
 - default field separator is whitespace
- A record is the collection of fields in a line
- A data file is made up of records

Example Input File

	Field 1 (First_Name)	Field 2 (Last_Name)	Field 3 (Pay_Rate)	Field 4 (Hours)
Record 2	Susan	White	6.00	23
	Mark	Eagle	6.25	40
Record 4	Tuan	Nguyen	7.89	44
	Dan	Black	7.23	40
	Amanda	Trapp	6.95	40
	Brian	Devaux	7.95	0
	Chris	Walljasper	6.89	32
	Mary	Lamb	8.22	40
Record 10	Jackie	Kammaoto	7.59	40
	Nicky	Barber	6.35	40

A file with 10 records, each with four fields

Buffers



- awk supports two types of buffers:
record and field
- field buffer:
 - one for each fields in the current record.
 - names: \$1, \$2, ...
- record buffer :
 - \$0 holds the entire record

Some System Variables

FS	Field separator (default=whitespace)
RS	Record separator (default=\n)
NF	Number of fields in current record
NR	Number of the current record
OFS	Output field separator (default=space)
ORS	Output record separator (default=\n)
FILENAME	Current filename

Example: Records and Fields

```
% cat emps
```

Tom Jones	4424	5/12/66	543354
Mary Adams	5346	11/4/63	28765
Sally Chang	1654	7/22/54	650000
Billy Black	1683	9/23/44	336500

```
% awk '{print NR, $0}' emps
```

1	Tom Jones	4424	5/12/66	543354
2	Mary Adams	5346	11/4/63	28765
3	Sally Chang	1654	7/22/54	650000
4	Billy Black	1683	9/23/44	336500



Example: Colon as Field Separator

```
% cat em2
```

```
Tom Jones:4424:5/12/66:543354
```

```
Mary Adams:5346:11/4/63:28765
```

```
Sally Chang:1654:7/22/54:650000
```

```
Billy Black:1683:9/23/44:336500
```

```
% awk -F: '/Jones/{print $1, $2}' em2
```

```
Tom Jones 4424
```

Pattern types

- match
 - entire input record
regular expression enclosed by ‘/’ s
 - explicit pattern-matching expressions
~ (match), !~ (not match)
- expression operators
 - arithmetic
 - relational
 - logical



Example: match input record

```
% cat employees2
```

```
Tom Jones:4424:5/12/66:543354
```

```
Mary Adams:5346:11/4/63:28765
```

```
Sally Chang:1654:7/22/54:650000
```

```
Billy Black:1683:9/23/44:336500
```

```
% awk -F: '/00$/ ' employees2
```

```
Sally Chang:1654:7/22/54:650000
```

```
Billy Black:1683:9/23/44:336500
```

- Further Reading:

www.hcs.harvard.edu/~dholland/computers/awk.html

<https://www.digitalocean.com/community/tutorials/how-to-use-the-awk-language-to-manipulate-text-in-linux>

sed

- a programming language designed for text processing
- Used for processing regular expressions in a script
- Used when the text is in a stream (no delimited field structure)
- typically used as a stream editor (thus the name)
- a powerful standard feature of most Unix-like operating systems

sed operations:

- Loops through a file line by line
- Looks for text patterns
- Executes commands on a match
 - Substitute, delete, insert a new line, etc.

Useful for:

- Transforming data files
- Finding text strings and changing them

Programming constructs:

- A rather primitive language

- Further Reading:

<http://www.wikiwand.com/en/Sed>

https://www.tutorialspoint.com/sed/sed_overview.htm