CSCI 3308 – Software Development Methods and Tools

University of Colorado
Boulder

Reminders

Topics

- Regular Expressions (regexes)
- awk
- sed
- Shell scripting

Regular Expressions

- Regular expressions allow you to specify a certain pattern of text within a text document.
- Use letters and symbols to define a pattern that's searched for.
- Example
  - In Windows, a wildcard notation such as *.txt finds all text files in a directory.
  - The regex equivalent is ^.*\.txt$

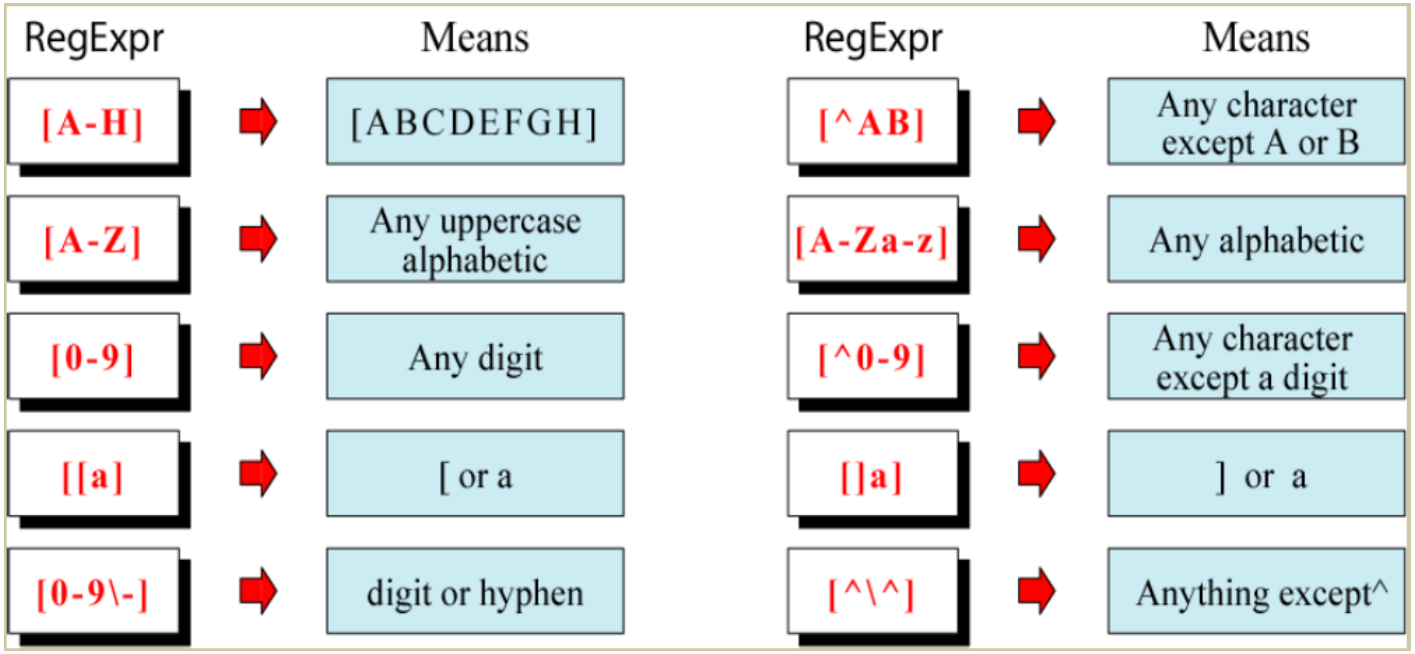
Regular Expressions

Regular expression	Description	Example	Type
*	Matches 0 or more occurrences of the previous character	letter* matches lette, letter, letterr, letterrrr, letterrrrr, and so on	Common
?	Matches 0 or 1 occurrences of the previous character	letter? matches lette, letter	Extended
+	Matches 1 or more occurrences of the previous character	letter+ matches letter, letterr, letterrrr, letterrrrr, and so on	Extended
. (period)	Matches 1 character of any type	letter. matches lettera, letterb, letterc, letter1, letter2, letter3, and so on	Common
[...]	Matches one character from the range specified within the braces	letter[1238] matches letter1, letter2, letter3, and letter8; letter[a-c] matches lettera, letterb, and letterc	Common
[^...]	Matches one character NOT from the range specified within the braces	letter[^1238] matches letter4, letter5, letter6, lettera, letterb, and so on (any character except 1, 2, 3, or 8)	Common
{}	Matches a specific number or range of the previous character	letter{3} matches letterrr letter, whereas letter {2,4} matches letterr, letterrr, and letterrrr	Extended
^	Matches the following characters if they are the first characters on the line	^letter matches letter if letter is the first set of characters in the line	Common
$	Matches the previous characters if they are the last characters on the line	letter$ matches letter if letter is the last set of characters in the line	Common
(... | ...)	Matches either of two sets of characters	(mother|father) matches the word “mother” or “father”	Extended

Regular Expressions

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 a string
$	Anchor – The end of the string
\	Escape character
|	Boolean OR
{m,n}	The preceding character appears **m** to **n** times
\b	Anchor – The beginning of a word
[[:blank:]]	Space or Tab

Regular Expressions



Regular Expressions

^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.

grep with regex

- The most common way to search for information using regular expressions is the grep command (global regular expression print).

- List only those lines in the file project4 that contain the words “Algonquin Park”

```
[root@server1 ~]# grep "Algonquin Park" project4
room this year while vacationing in Algonquin Park - I
I have been reading on the history of Algonquin Park but
[root@server1 ~]#_
```

grep with regex

- To return the lines that do not contain the text "Algonquin Park," you can use the –v option of the grep command to reverse the meaning of the previous command:

```
[root@server1 ~]# grep -v "Algonquin Park" project4
Hi there, I hope this day finds you well.

Unfortunately, we were not able to make it to your dining
especially wished to see the model of the Highland Inn
and the train station in the dining room.

nowhere could I find a description of where the Highland
Inn was originally located on Cache Lake.

If it is no trouble, could you kindly let me know such that
I need not wait until next year when I visit your lodge?

Regards,
Mackenzie Elizabeth
[root@server1 ~]#_
```

grep with regex

- Keep in mind that the text being searched is case sensitive; to perform a case-insensitive search, use the –i option to the grep command:

```
[root@server1 ~]# grep "algonquin park" project4
[root@server1 ~]#_
[root@server1 ~]# grep -i "algonquin park" project4
room this year while vacationing in Algonquin Park - I
I have been reading on the history of Algonquin Park but
[root@server1 ~]#_
```

grep with regex

- To view lines that start with the word “I,” you can enter the following command:

```
[root@server1 ~]# grep "^I " project4
I have been reading on the history of Algonquin Park but
I need not wait until next year when I visit your lodge?
[root@server1 ~]#_
```

egrep with regex

- To view lines that contain the text “lodge” or “Lake,” you need to use an extended regular expression and the egrep command, as follows:

```
[root@server1 ~]# egrep "(lodge|Lake)" project4
Inn was originally located on Cache Lake.
I need not wait until next year when I visit your lodge?
[root@server1 ~]#_
```

egrep with regex

- We want to look for names that start with “T,” are followed by at least one, but no more than two, consecutive vowels, and end in “m.”

```
dave@howtogeek:~$ grep -E 'T[aeiou]{1,2}m' geeks.txt
Tim Brooks
Tom Westrick
Team Geek
dave@howtogeek:~$
```

grep – Additional Examples

- Literal matches
  - grep -ivn “unix” textfile.txt
- Anchor matches
  - grep “^unix” textfile.txt
- Matching Any Character
  - grep “..eat” textfile.txt
- Bracket Expressions
  - grep “^[A-Z]” textfile.txt
- Repeat Pattern Zero or More Times
  - grep “[A-Za-z]*” textfile.txt

awk

- The **awk command** searches for patterns of text and performs some action on the text it finds. The awk command treats each line of text as a record in a database and each word in a line as a database field.
- For example, the line "Hello, how are you?" has four fields: "Hello," "how," "are," and "you?" These fields can be referenced in the awk command using $1, $2, $3, and $4.
- Note: awk supports two types of buffers: record and field
  - Fields buffer:
    - ***One for each fields in the current record***
    - ***Names: $1, $2, ..***
  - Record buffer:
    - ***$0 holds the entire record***

awk

- For example, to display only the first and fourth words on lines of the prologue file that contain the word "the," one can use the following command.

```
[root@serverl -]# cat prologue l awk '/the/ {print $1, $4}'
From fatal
A star-cross'd
Do death
The of
And of
Which, children's
Is two
[root@serverl -]#_
```

awk

- Most configuration files on Linux systems are delimited using colon (:) characters. To change the delimiter that awk uses, you can specify the –F option to the command. For example, the following example lists the last 10 lines of the colon-delimited file /etc/passwd and views only the sixth and seventh fields for lines that contain the word “bob” in the last 10 lines of the file:

```
[root@server1 ~]# tail /etc/passwd
news:x:9:13:News server user:/etc/news:/sbin/nologin
smolt:x:490:474:Smolt:/usr/share/smolt:/sbin/nologin
backuppc:x:489:473::/var/lib/BackupPC:/sbin/nologin
pulse:x:488:472:PulseAudio System Daemon:/var/run/pulse:/sbin/nologin
gdm:x:42:468::/var/lib/gdm:/sbin/nologin
hsqldb:x:96:96::/var/lib/hsqldb:/sbin/nologin
jetty:x:487:467::/usr/share/jetty:/bin/sh
bozo:x:500:500:bozo the clown:/home/bozo:/bin/bash
bob:x:501:501:Bob Smith:/home/bob:/bin/bash
user1:x:502:502:sample user one:/home/user1:/bin/bash
[root@server1 ~]# tail /etc/passwd | awk -F : '/bob/ {print $6, $7}'
/home/bob /bin/bash
[root@server1 ~]#_
```

awk – System Variables

FS – Field Separator (default = whitespace)

RS – Record Separator (default = \n)

NF – Number of Fields in the current record

NR – Number of the current record

OFS – Output file separator (default = space)

ORS – Output record separator (default = \n)

FILENAME - Current FileName

awk – Additional Examples

```
awk '{print}' employee.txt
```

```
awk '/manager/ {print}' employee.txt
```

```
awk '{print $1,$4}' employee.txt
```

```
awk '{print NR,$0}' employee.txt
```

```
awk '{print $1,$NF}' employee.txt
```

```
awk 'NR==3, NR==6 {print NR,$0}' employee.txt
```

```
awk '$2 !~ /manager/ {print}' employee.txt
```

```
awk '$2 ~ /manager/ {print}' employee.txt
```

```
$ awk 'BEGIN { for(i=1;i<=6;i++) print "square of", i, "is",i*i; }' employee.txt
```

sed

- The **sed command** is typically used to search for a certain string of text and replaces that text string with another text string using the syntax s/search/replace/

- For example, the following output demonstrates how to use sed to search for the string “the” and replace it with the string “THE” in the prologue file.

[root@server1 -]# **cat prologue I sed s/ the/THE/**

Two households, both alike in dignity,
In fair Verona, where we lay our scene,
From ancient grudge break to new mutiny,
Where civil blood makes civil hands unclean.
From forth THE fatal loins of these two foes
A pair of star-cross'd lovers take THEir life;
Whole misadventured piteous overthrows
Do with THEir death bury their parents' strife.
The fearful passage of THEir death-mark'd love,
And THE continuance of their parents' rage,
Which, but THEir children's end, nought could remove,
Is now THE two hours' traffic of our stage;
The which if you with patient ears attend,
What here shall miss, our toil shall strive to mend.

*Notice from the preceding output that sed only searched for and replaced the first occurrence of the string “the” in each line.

sed

- To have sed globally replace all occurrences of the string “the” in each line, append a g to the search-and-replace expression:

- For example, the following output demonstrates how to use sed to search for the string “the” and replace it with the string “THE” in the prologue file.

```
[root@server1 ~]# cat prologue | sed s/the/THE/g

Two households, both alike in dignity,
In fair Verona, where we lay our scene,
From ancient grudge break to new mutiny,
Where civil blood makes civil hands unclean.
From forth THE fatal loins of THEse two foes
A pair of star-cross'd lovers take THEir life;
Whole misadventured piteous overthrows
Do with THEir death bury THEir parents' strife.
The fearful passage of THEir death-mark'd love,
And THE continuance of THEir parents' rage,
Which, but THEir children's end, nought could remove,
Is now THE two hours' traffic of our stage;
The which if you with patient ears attend,
What here shall miss, our toil shall strive to mend.
[root@server1 ~]#_
```

sed

- You can also tell sed the specific lines to search by prefixing the search-and-replace expression. For example, to force sed to replace the string “the” with “THE” globally on lines that contain the string “love,” you can use the following command:

```
[root@server1 ~]# cat prologue | sed /love/s/the/THE/g
Two households, both alike in dignity,
In fair Verona, where we lay our scene,
From ancient grudge break to new mutiny,
Where civil blood makes civil hands unclean.
From forth the fatal loins of these two foes
A pair of star-cross'd lovers take THEir life;
Whole misadventured piteous overthrows
Do with their death bury their parents' strife.
The fearful passage of THEir death-mark'd love,
And the continuance of their parents' rage,
Which, but their children's end, nought could remove,
Is now the two hours' traffic of our stage;
The which if you with patient ears attend,
What here shall miss, our toil shall strive to mend.
[root@server1 ~]#_
```

sed

- You can also force sed to perform a search-and-replace on certain lines only. To replace the string “the” with “THE” globally on lines 5 to 8 only, you can use the following command:

```
[root@server1 ~]# cat prologue | sed 5,8s/the/THE/g
Two households, both alike in dignity,
In fair Verona, where we lay our scene,
From ancient grudge break to new mutiny,
Where civil blood makes civil hands unclean.
From forth THE fatal loins of THEse two foes
A pair of star-cross'd lovers take THEir life;
Whole misadventured piteous overthrows
Do with THEir death bury THEir parents' strife.
The fearful passage of their death-mark'd love,
And the continuance of their parents' rage,
Which, but their children's end, nought could remove,
Is now the two hours' traffic of our stage;
The which if you with patient ears attend,
What here shall miss, our toil shall strive to mend.
[root@server1 ~]#_
```

sed

- You can also use sed to remove unwanted lines of text. To delete all the lines that contain the word "the," you can use the following command:

```
[root@server1 ~]# cat prologue | sed /the/d
Two households, both alike in dignity,
In fair Verona, where we lay our scene,
From ancient grudge break to new mutiny,
Where civil blood makes civil hands unclean.
Whole misadventured piteous overthrows
The which if you with patient ears attend,
What here shall miss, our toil shall strive to mend.
[root@server1 ~]#_
```

Shell Scripting

- Shell scripting allows you to create automated tasks.
- There are different types of shells in Linux. For example:
  - BASH shell (Bourne-Again SHell – default command-line interface in Linux)
  - Korn shell
  - C shell
- The handling of variables can vary between the shells.

- root@ashraf:/# echo $SHELL
  /bin/bash

- mkdir /scripts
- cd /scripts
- vi script1.sh

Shell Scripting - Example

```
#**************************************
#!/bin/bash
#**************************************
#Display a message

clear

echo "Hello World!“
```

- To run the script, use:

./script1.sh

- May need to set the script file to have execute privileges so use:

chmod +x script1.sh

Shell Scripting – Decisions

- Decision construct

```
if this is true
then
do these commands
elif this is true
then
do these commands
else
do these commands
fi
```

- Example

```
[root@server1 ~]# cat myscript
#!/bin/bash
echo -e "Today's date is: \c"
date
echo -e "\nThe people logged into the system include:"
who
echo -e "\nWould you like to see the contents of /?(y/n)-->\c"
read ANSWER
if [ $ANSWER = "y" ]
then
echo -e "\nThe contents of the / directory are:"
ls -F /
fi
```

Shell Scripting – Test Stmts

Test statement	Returns true if:
[A = B]	String A is equal to String B.
[A != B]	String A is not equal to String B.
[A -eq B]	A is numerically equal to B.
[A -ne B]	A is numerically not equal to B.
[A –lt B]	A is numerically less than B.
[A –gt B]	A is numerically greater than B.
[A –le B]	A is numerically less than or equal to B.
[A –ge B]	A is numerically greater than or equal to B.
[-r A]	A is a file/directory that exists and is readable (r permission).
[-w A]	A is a file/directory that exists and is writable (w permission).
[-x A]	A is a file/directory that exists and is executable (x permission).
[-f A]	A is a file that exists.
[-d A]	A is a directory that exists.

Questions?