Software I: Utilities and Internals

Lecture 4 – Regular Expressions, **`grep`** and **`sed`**

What Is A Regular Expression?

- A ***regular expression*** is a pattern consisting of a sequence of characters that is matched against text.
- Regular expressions give us a way of recognizing words, numbers and operators that appear as part of a larger text so the computer can process them in a meaningful and intelligent way.

What are Atoms?

- Regular expressions consist of atoms and operators.
- An ***atom*** specifies what text is to be matched and where it can be found.
- There are five types of atoms that can be found in text:
  - Single characters
  - Dots
  - Classes
  - Anchors
  - Back references

Single Characters

- The most basic atom is a single character; when a single character appears in a regular expression, that character must appear in the text for there to be a successful match.
- Example (String is "**`Hello`**"; Regular Expression is "**`l`**")
  - The match is successful because "**`l`**" appears in "**`Hello`**"
  - If there regular expression had been "**`s`**", there would be no match.

Dot

- A ***dot*** ("`.`") matches any character except new line ('`\n`').
- Example
  - `a.` matches `aa`, `ab`, `ac`, `ad`, `aA`, `aB`, `a3`, etc.
  - `.` will match any character in `HELLO`, `H.` will match the `HE` in `HELLO`, `h.` matches nothing in `HELLO`.

Class

- A class consists of a set of ASCII character, any one of which matches any character in the text.
- Classes are written with the set of characters contained within brackets.
- Example
- `[ABL]` matches either "`L`" in `HELLO`.

Ranges and Exceptions in Classes

- A range of characters can be used in a class:
  - `[a-d]` or `[A-Za-z]`
- Sometimes is it easier to specify what characters DON'T appear. This is done using exclusion (`^`).
- Examples
  - `[^aeiou]` specifies anything but a vowel.
  - `[^0-9]` specfies anything but a digit.

Classes – Some Examples

Regular Expression	Means
`[A-H]`	`[ABCDEFGH]`
`[A-Z]`	Any uppercase letter
`[0-9]`	any digit
`[[a]`	`[` or `a`
`[0-9\-]`	digit or hyphen
`[^AB]`	Any character except `A` or `B`
`[A-za-z]`	Any letter
`[^0-9]`	Any character other than a digit
`[]a]`	`]` or `a`
`[^\^]`	Anything but `^`

Anchors

- Anchors line up the pattern with a particular part of the string:
  - ^ Beginning of the line
  - $ End of the line
  - \< Beginning of a word
  - \> End of a word

Anchors- Examples

- Sample text: **`One line of text\n`**
- **`^One`** Matches
- **`text$`** Matches
- **`\<line`** Matches
- **`\>line`** Does not match
- **`f\>`** Matches

What are Operators?

- Operators provide us with a way to combine atoms to form larger and more powerful regular expressions.
- Operators play the same role as mathematical operators play in algebraic expressions.
- There are five types of operators that can be found in text:
  - Sequence
  - Alternation
  - Repetition
  - Group
  - Save

Sequence

- No symbol is used for the sequence operator; all you need is to have two atoms appear in sequence.
- We can match the string CHARACTER with the pattern ACT because we find the sequence ACT in our string.

Sequence - Examples

- **`dog`** – matches the character sequence "**`dog`**"
- **`a..b`** – matches **`a`**, any two characters, then **`b`**
- **`[2-4][0-9]`** – matches a number between **`20`** and **`49`**.
- **`^$`** - matches a blank line
- **`^.$`** - matches a line with only one character
- **`[0-9] - [0-9]`** – matches two digits with a dash in between.

Alternation

- The alternation operator (**`|`**) defines one or more alternatives, either of which can appear in the string.
- Examples
  - **`UNIX|unix`** matches either **`UNIX`** or **`unix`**
  - **`Ms|Mrs|Miss`** matches **`Ms`**, **`Mrs`** or **`Miss`**
  - **`FE|EL`** matches **`HELLO`** because one of the alternatives matches it.

Repetition

- Repetition refers to a definite or indefinite number of times that one or more characters can appear.
- The most common forms of repetition use three "short form" repetition operators:
- **`*`** - zero or more occurrences
- **`+`** - one or more occurrences
- **`?`** - zero or one occurrences

`*` - Examples

- **`BA*`** - **`B`**, **`BA`**, **`BAA`**, **`BAAA`**, **`BAAAA`**
- **`B.*`** - **`B`**, **`BA`**, **`BB`**, **`BC`**, **`BD`**, …, **`BAA`**, **`BAB`**, **`BAC`**, …
- **`.*`** - any sequence of zero or more characters

+ - Examples

- **BA+** - **BA**, **BAA**, **BAAA**, **BAAA**, …
- **B.+** - **BA**, **BB**, **BC**, **BD**, …, **BZ**, **BAA**, **BAB**, …
- **.+**- any sequence of one or more characters

? - Examples

- **d?** - zero or one d
- **[0-9]?** – zero or one digit
- **[^A-Z]?** – zero or one character except a capital letter
- **[A-Za-z]?** – zero or one letter

General Cases of Repetition

- Repetition can be stated in more general terms using a set of escaped brackets containing two numbers separated by a comma
- Example
  - `B\{2, \5}` would match **BB**, **BBB**, **BBBB**, **BBBBB**
- The minimum or maximum value can be omitted:
  - `CA\{5\}` matches **CAAAAA**
  - `CA\{2, \}` matches **CAA**, **CAA**, **CAAA**,…
  - `CA \{, 5\}` matches **CA**, **CAA**, **CAAA**, **CAAAA**, **CAAAAA**

Group Operator

- The group operator is a pair of parentheses around a group of characters, causing the next operator to apply to the group, not just a single character:
- Example
  - `AB*C` - matches **AC**, **ABC**, **ABBC**, **ABBC**, …
  - `(AB)*C` – matches **C**, **ABC**, **ABABC**, **ABABC**, …

What is **grep**?

- **grep** (general regular expression program) allows the user to print each line in a text file that contains a particular pattern.

What is **grep**?

- The name **grep** stands for "***g***eneral ***r***egular ***e***xpression ***p***rogram."
- The general format is
  grep *pattern filenames*
- The input can be from files or from **stdin**.
  - **grep -n variable *.[ch]**
    prints every line in every c source file or header file containing the word *variable* (and prints a line number).

Examples of **grep**

grep From $MAIL

- *Print message headers in the mailbox*

grep From $MAIL | grep –v mary

- *which ones are* not *from Mary*

grep –y mary $HOME/lib/phone-book

- *Find Mary's phone-book.*

who | grep mary

- *Is Mary logged in?*

ls | grep –v temp

- *List all the files without* temp *in their name*

Options for **grep**

- **–i** - ignore case – treat upper and lower case the same.
- **–n** – provide line numbers
- **–v** - reverse – print lines without the pattern.
- **–c** – provide a count of the lines with the pattern, instead of displaying these lines.

grep Patterns

- **grep** patterns can be more complicated:
  - **grep c***
    0 or more occurrences of c in the pattern
  - **grep sieg* /etc/patterns**
    Check the password file for sie, sieg, siegg, sieggg, etc.
  - **grep [abc]**
    Check for an occurrence of any of these three characters.
  - grep [br]ob /etc/passwd
    Look for bob *or* rob *in the password file.*
  - grep [0-9]* hithere.c
    Look for numbers in the program.

^ And $ In A **grep** Pattern

- The metacharacters **^** and **$** anchor text to the beginning and end of lines, respectively:
  - **grep From $MAIL**
    Check mail for lines containing From
  - **grep '^From' $MAIL**
    Check mail for lines **beginning** *with From*
  - **grep ';$' hello.c**
    Display lines ending with **;**

Other Pattern Metacharacters

- A circumflex inside the brackets causes grep to reverse its meaning

  `grep [^0-9] hithere.c`

  Display the lines without digits

- A period represents any single character

  `ls -l | grep '^d'`

  List the subdirectories

  `ls -l | grep '^.......rw'`

  List files others can read and write (the seven dots are for the file type and other permissions)

*

- **`x*`** - 0 or more **`x`**s
- **`.*`** - 0 or more of any character
- **`.*x`** – anything followed by an **`x`**.
- **`xy*`** - **`x`** followed by zero or more **`y`**s

  The `*` applies to only one character.

  `xy`, **`xyy`**, **`xyyy`**, etc. NOT **`xy`**, **`xyxy`**, **`xyxyxy`**, etc.

`[a-zA-Z]*` - 0 or more letters

`[a-zA-Z][a-zA-Z]*` - 1 or more letters

grep – Some More Examples

- **grep '^[^:]*::' /etc/passwd**

  Lists users without a password – it looks from the beginning of the line for non-colons followed by two consecutive colons.
- **w –h | grep days**

  who without a heading – lists everyone who has been idle for more than 1 day.
- **w –h | grep days | cut –c1–8**

  cuts out some of the output (includes only columns 1 through 8)
- **grep –l float ***

  lists only the file names for the files in this subdirectory containing the string **float**.

grep – Some More Examples

```
[SIEGFRIE@panther c]$ cat > memo
data is correct before we publish it.
I thought you would have known by now.
[SIEGFRIE@panther c]$ grep –w now memo
I thought you would have known by now.
[SIEGFRIE@panther c]$ cat >errors
0[0–9].e[0–9]*
[SIEGFRIE@panther c]$ cat >sketch
00.e8
9/12
[SIEGFRIE@panther c]$ grep –f errors sketch
00.e8
[SIEGFRIE@panther c]$
```

grep Family

- The grep family includes 2 additional variations of grep:
- fgrep – fast grep uses only sequence of cahracters in a pattern, but works more quickly than grep.
- egrep – extended grep handles a wider array of regular expressions.

fgrep – Examples

```
SIEGFRIE@panther:~$ cat raven
Once upon a midnight dreary, while I pondered, weak and weary,
Over many a quaint and curious volume of forgotten lore.
   While I nodded, nearly napping, suddenly there came a tapping,
As of some one gently rapping, rapping at my chamber door.
..Tis some visiter,. I muttered, .tapping at my chamber door.
            Only this and nothing more..
… …
 And the Raven, never flitting, still is sitting, still is
sitting
On the pallid bust of Pallas just above my chamber door;
    And his eyes have all the seeming of a demon.s that is
dreaming,
    And the lamp-light o.er him streaming throws his shadow on
the floor;
And my soul from out that shadow that lies floating on the floor
            Shall be lifted.nevermore!
SIEGFRIE@panther:~$
```

```
SIEGFRIE@panther:~$ fgrep Raven raven
In there stepped a stately Raven of the saintly days
of yore;
Ghastly grim and ancient Raven wandering from the
Nightly shore.
            Quoth the Raven .Nevermore..
    But the Raven, sitting lonely on the placid
bust, spoke only
    But the Raven still beguiling all my fancy into
smiling,
            Quoth the Raven .Nevermore..
            Quoth the Raven .Nevermore..
            Quoth the Raven .Nevermore..
            Quoth the Raven .Nevermore..
    And the Raven, never flitting, still is sitting,
still is sitting
```

```
SIEGFRIE@panther:~$ fgrep –v Raven raven

Once upon a midnight dreary, while I pondered, weak and weary,
Over many a quaint and curious volume of forgotten lore.
    While I nodded, nearly napping, suddenly there came a
tapping,
As of some one gently rapping, rapping at my chamber door.
..Tis some visiter,. I muttered, .tapping at my chamber door.
            Only this and nothing more..
… …
And my soul from out that shadow that lies floating on the floor
            Shall be lifted.nevermore!

SIEGFRIE@panther:~$
```

egrep

```
[SIEGFRIE@panther c]$ cat alphvowels
^[^aeiou]*a[^aeiou]*e[^aeiou]*o[^aeiou]*u [^aeiou]*$
[SIEGFRIE@panther c]$ egrep -f alphvowels dict | 3
abstemious  abstemious  abstentious
achelious   acheirous   acleistous
affectious  annelidous  arsenous
… …
```

- **egrep** extends the capabilities with three additional metacharacters: **? + |**
  - *r***+** - 1 or more occurrences of *r*
  - *r***?** – 0 or more occurrences of *r*
  - *r1* **|** *r2* – Either *r1* or *r2*
- **egrep 'cookie|donut' oreo**

Searching for File Content

```
SIEGFRIE@panther:~$ ls | grep flea
fleas
fleass
fleast
fleawrite
newfleas
SIEGFRIE@panther:~$ ls | grep 'fl*'
160L2Handout.pdf
160l4notes.pdf
270cl1.pdf
binfile.c
BlindOpportunities.pdf
filename
```

```
final
find
fl
fleas
fleass
fleast
fleawrite
myfile
mystuff
newfleas
test.f
under.f
yourstuff
SIEGFRIE@panther:~$
```

Searching for Files

```
SIEGFRIE@panther:~$ ls | grep flea
fleas
fleass
fleast
fleawrite
newfleas
SIEGFRIE@panther:~$ ls | grep 'fl*'
160L2Handout.pdf
160l4notes.pdf
270cl1.pdf
binfile.c
BlindOpportunities.pdf
filename
```

`sed`

What Are Filters?

- A filter is a UNIX program that reads input (usually **`stdin`**), performs some transformation on it and writes it (usually to **`stdout`**).
- This follows the UNIX/Linux model of building simple components and then combining them to create more powerful applications.
  - We might use **`grep`** or **`tail`** to select some of our input, **`sort`** to sort it, **`wc`** to count characters and/or lines, etc.

sed – The Stream Editor

- The basic command is:
  sed 'list of editing commands' filename
- **sed** does not alter the input file unless output is redirect there. This
  sed '…' file >file
  is a really bad idea; it is much better to store the results in a temporary file.
- sed outputs each line automatically, so no **p**(rint) commands are necessary(unless you are using the **-n** option.

sed – The Stream Editor

```
[SIEGFRIE@panther ~]$ sed 's/knees/feet/g' <mystuff\
      >yourstuff
[SIEGFRIE@panther ~]$ cat mystuff
  This is a test of the emergency programming system.  If
  this were a real emergency, bend forward and kiss your
  knees goodbye
[SIEGFRIE@panther ~]$ cat yourstuff
  This is a test of the emergency programming system.  If
  this were a real emergency, bend forward and kiss your
  feet goodbye
[SIEGFRIE@panther ~]$
```

sed patterns

- **sed** patterns almost always need quotes because their metacharacters usually have a special meaning to the shell.
- **du** – estimate disk usage
- **du –a** – include files as well as directories.

du and **sed** – An Example

```
[SIEGFRIE@panther ~]$ du -a ch2.*
4       ch2.1
4       ch2.2
4       ch2.3
4       ch2.4
4       ch2.5
[SIEGFRIE@panther ~]$ du -a ch2.* | sed 's/.*c/c/'
ch2.1
ch2.2
ch2.3
ch2.4
ch2.5
[SIEGFRIE@panther ~]$
```

who And **sed**

```
[SIEGFRIE@panther ~]$ who
SIEGFRIE pts/2 Dec  8 18:16 (pool-98-….verizon.net)
don      pts/4 Nov 25 10:07 (10.80.4.78)
CHRISTOP pts/5 Dec  8 16:32 (pool-141-….verizon.net)
[SIEGFRIE@panther ~]$ who | sed 's/ .* / /'
SIEGFRIE (pool-98-….verizon.net)
don (10.80.4.78)
CHRISTOP (pool-141-….verizon.net)
[SIEGFRIE@panther ~]$
```

who am I And **sed**

```
[SIEGFRIE@panther ~]$ who am i
SIEGFRIE pts/2  Dec  8 18:16 (pool-98-….verizon.net)
[SIEGFRIE@panther ~]$ getname
SIEGFRIE
[SIEGFRIE@panther ~]$
```

What are scripts?

- A script is a series of editing commands placed in a file that can be in a **`sed`** command.
- Example

```
SIEGFRIE@panther:~$ cat fleas
Great fleas have little fleas
 upon their backs to bite 'em
And little fleas have lesser fleas
 and so on ad infinitum.
And the great fleas themselves, in turn,
 have greater fleas to go on;
While these again have greater still,
 and great still and so on.
```

`ind`

- We are going to implement another filter called **`ind`** which will indent its input one tab stop.
- Our initial implementation is

  `sed 's/^/→/' $*`

  This places tabs on lines that would otherwise be blank.
- We can avoid this problem by writing

  `sed '/^$/!s/^/→/' $*`

  It substitutes on all lines EXCEPT those with no content.

ind2

```
[SIEGFRIE@panther ~]$ cat bin/ind2
sed 's/^/       /
     3q'
[SIEGFRIE@panther ~]$ sed 3q fleas
Great fleas have little fleas
 upon their backs to bite 'em
And little fleas have lesser fleas
[SIEGFRIE@panther ~]$ cat fleas | ind2
        Great fleas have little fleas
         upon their backs to bite 'em
        And little fleas have lesser fleas
[SIEGFRIE@panther ~]$
```

sed Commands from Files

- **sed** commands can be taken from files by writing:
  sed –f cmdfile
- Number selectors can now be used for printing, deleting and substituting.

sed -f – An Example

```
[SIEGFRIE@panther ~]$ cat fleawrite
2p
4p
6p
8p
[SIEGFRIE@panther ~]$ sed -f fleawrite fleas
Great fleas have little fleas
 upon their backs to bite 'em
 upon their backs to bite 'em
And little fleas have lesser fleas
 and so on ad infinitum.
 and so on ad infinitum.
And the great fleas themselves, in turn,
And the great fleas themselves, in turn,
 have greater fleas to go on;
While these again have greater still,
While these again have greater still,
 and great still and so on.
```

repeated text

sed -n -f – An Example

- sed –n suppresses the automatical printing

```
[SIEGFRIE@panther ~]$ cat fleas | \
                      sed -n -f fleawrite
 upon their backs to bite 'em
 and so on ad infinitum.
And the great fleas themselves, in turn,
While these again have greater still,
[SIEGFRIE@panther ~]$
```

sed And Patterns

- **sed '/pattern/q'**
  Prints input up to the pattern and then quits.
- **sed '/pattern/d'** *Deletes every line with the pattern*
- **sed –n '/pattern/p'**
  Prints every line with the pattern
- **sed –n '/pattern/!p'**
  Prints every line without the pattern
- **sed 's/$/**
  /' *Inserts newlines*
- **sed 's/[→][→]*/**
  /g' *Replaces each string of blanks and tabs with a newline (splits input into one word per line)*

```
SIEGFRIE@panther:~$ cat script.sed
#This line is a comment
s/fleas/Fleas/g
6,8s/ on/ on and on/g
SIEGFRIE@panther:~$ sed –f script.sed <fleas >fleas2
SIEGFRIE@panther:~$ more fleas2
Great Fleas have little Fleas
 upon their backs to bite 'em
And little Fleas have lesser Fleas
 and so on ad infinitum.
And the great Fleas themselves, in turn,
 have greater Fleas to go on and on;
While these again have greater still,
 and great still and so on and on.
SIEGFRIE@panther:~$
```

Instruction Format

- Each instruction is of the format:

  address or line number(s)

  ! for complement is optional

  command

- Examples

```
2, 14 s/A/B
30d
42d
```

sed – An Example

```
SIEGFRIE@panther:~$ cat hello.dat
Hello friends
Hello guests
Hello students
Welcome
SIEGFRIE@panther:~$ cat hello.sed
1, 3s/Hello/Greetings/
2,3s/friends/buddies/
SIEGFRIE@panther:~$ sed -f hello.sed hello.dat
Greetings friends
Greetings guests
Greetings students
Welcome
SIEGFRIE@panther:~$
```

Commands

- There are nine(?) categories of commands in sed:
  1. Line number
  2. Modify
  3. Substitute
  4. Transform
  5. Input/Output
  6. Fies
  7. Branch
  8. Hold Space
  9. Quit

Line number

- The line number command (`=`) write the line number at the beginning of the line when it writes the line to output.
- It does NOT affect the pattern space (a buffer holding one line of text).
- It is similar to **`grep -n`**.

Line Number - Example

```
SIEGFRIE@panther:~$ cat fleas | sed '='
1
Great fleas have little fleas
2
 upon their backs to bite 'em
… …
6
 have greater fleas to go on;
7
While these again have greater still,
8
 and great still and so on.
SIEGFRIE@panther:~$
```

Modify

- Modify commands are are to insert, append, change, or delete one or more whole lines.
- Any text associated with a modify command must be placed on the line after the command.

`i` (Insert) and `a` (Append)

- Insert adds one or more line of text directly to the output ***before*** the address.
- Append adds one or more line of text directly to the output ***after*** the address.
- These lines are written directly to standard output and are never in the pattern space.

`i` – An Example

```
SIEGFRIE@panther:~$ cat fleas | sed '1i\
Fleas\
by Ima Canine\'
Fleas\
by Ima Canine
Great fleas have little fleas
 upon their backs to bite 'em
And little fleas have lesser fleas
 and so on ad infinitum.
And the great fleas themselves, in turn,
 have greater fleas to go on;
While these again have greater still,
 and great still and so on.
SIEGFRIE@panther:~$
```

`a` – An Example

```
SIEGFRIE@panther:~$ cat fleas | sed '1,3a\
> ----------------------------------------\
> '
Great fleas have little fleas
----------------------------------------

 upon their backs to bite 'em
----------------------------------------

And little fleas have lesser fleas
----------------------------------------

 and so on ad infinitum.
… …
While these again have greater still,
 and great still and so on.
SIEGFRIE@panther:~$
```

`c` - Change

- Change replaces a matched line with new text.
- Unlike insert and append, it accepts addresses in a variety of forms.

`c` – An Example

```
SIEGFRIE@panther:~$ cat fleas | sed '3c\
> And little fleas have little pests\'
Great fleas have little fleas
 upon their backs to bite 'em
And little fleas have little pests
 and so on ad infinitum.
And the great fleas themselves, in turn,
 have greater fleas to go on;
While these again have greater still,
 and great still and so on.
SIEGFRIE@panther:~$
```

`s` - Substitute

- The substitute command replaces text that is selected by a regular expression with a replacement string.
- It is essentially the same as search and replace in a word processor or text editor.
- The regular expressions that you use can contain characters, dot, class, anchors, sequences, and repetition.

s – An Example

```
SIEGFRIE@panther:~$ cat fleas | sed 's/fleas/bugs/'
Great bugs have little fleas
 upon their backs to bite 'em
And little bugs have lesser fleas
 and so on ad infinitum.
And the great bugs themselves, in turn,
 have greater bugs to go on;
While these again have greater still,
 and great still and so on.
SIEGFRIE@panther:~$
```

s – An Example

```
SIEGFRIE@panther:~$ cat fleas | sed
's/1s/fleas/bugs/g'
Great bugs have little bugs
 upon their backs to bite 'em
And little fleas have lesser fleas
 and so on ad infinitum.
And the great fleas themselves, in turn,
 have greater fleas to go on;
While these again have greater still,
 and great still and so on.
SIEGFRIE@panther:~$
```

s – An Example

```
SIEGFRIE@panther:~$ cat fleas | sed
'/fleas/s//bugs/g'
Great bugs have little bugs
 upon their backs to bite 'em
And little bugs have lesser bugs
 and so on ad infinitum.
And the great bugs themselves, in turn,
 have greater bugs to go on;
While these again have greater still,
 and great still and so on.
SIEGFRIE@panther:~$
```

s – An Example

```
SIEGFRIE@panther:~$ cat fleas | sed 's/fleas//'
Great  have little fleas
 upon their backs to bite 'em
And little  have lesser fleas
 and so on ad infinitum.
And the great  themselves, in turn,
 have greater  to go on;
While these again have greater still,
 and great still and so on.
SIEGFRIE@panther:~$ SIEGFRIE@panther:~$
```

`y` - Transform

- The Transform command (`y`) is used to change a character from one set of characters to a character from another set of characters.
- It is useful when encoding text or converting between ASCII and EBCDIC.

`y` – An Example

```
SIEGFRIE@panther:~$ cat fleas| sed '1y/aeiou/EIOUA/'
GrIEt flIEs hEvI lOttlI flIEs
 upon their backs to bite 'em
And little fleas have lesser fleas
 and so on ad infinitum.
And the great fleas themselves, in turn,
 have greater fleas to go on;
While these again have greater still,
 and great still and so on.
SIEGFRIE@panther:~$
```

y – An Example

```
SIEGFRIE@panther:~$ cat fleas |
> sed
'y/abcdefghijklmnopqrstuvwxyz/nopqrstuvwxyzabcdefghi
jklm/'
Gerng syrnf unir yvggyr syrnf
 hcba gurve onpxf gb ovgr 'rz
Aaq yvggyr syrnf unir yrffre syrnf
 naq fb ba nq vasvavghz.
Aaq gur terng syrnf gurzfryirf, va ghea,
 unir terngre syrnf gb tb ba;
Wuvyr gurfr ntnva unir terngre fgvyy,
 naq terng fgvyy naq fb ba.
SIEGFRIE@panther:~$
```