CEN 308
Lab Session
Week 9

Regular Expressions

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 ***<u>atom</u>*** 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
- line\> Matches
- **`f\>`** Matches

```
PEPPER@panther:~/270$ grep 'line\>' vitest1
one line of text
```

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`**, **`BAAAA`**, …
- **`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`**

(escape so the braces are interpreted as char)

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`**, **`ABBBC`**, …
  - **`\(AB\)*C`** – matches **`C`**, **`ABC`**, **`ABABC`**, **`ABABABC`**, …

(escape so the parentheses are interpreted as char)

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 –i 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, siegggg, 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
```

- 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`.