Chapter 3

Searching for regular expressions

Suppose that, rather than looking for fixed strings, we want to look for "patterns". Examples might be

- all words beginning with "`a`" and ending in "`ing`".
- all words that start with an "`s`", end with an "`s`" and have an "`ss`" in between.
- all lines with a double vowel immediately followed by a double consonant.

The `grep` and `egrep` commands provide a syntax for describing these expressions that is common to many Unix commands including `sed` and `awk` which we will meet later in this course.

The name given to one of these syntactic expressions used to describe a pattern is "regular expression".

3.1 Selecting one character from a set

Suppose we wanted to match those lines in a file containing either of the strings "`hack`" or "`Hack`". It is not enough to use the `-i` flag because then `grep` would match lines with expressions like "`HACK`" or "`hAcK`" as well.

We need a syntax to say "match a string whose first character is either an '`h`' or an '`H`', followed by '`ack`'". To do this, we use the expression "`[Hh]ack`" as our search string:

```
$ grep '[Hh]ack' wizard
Hackerly Spirit in numerous pleasing ways), but
_The_New_Hacker's_Dictionary_ (Eric S. Raymond,
Student, but the Divine Foolery of the Hacker.
Hack, then; strive against Mighty Problems, have
Right-Thinking Hacker is to rewrite the UNIX
of those who came before thee.  Hack, and hack
best hacks to the Net and gain in Repute
until Hackers respect thy Power, and other
```

The square brackets expression represents "any one character from the characters listed between the square brackets".

Note that square brackets have a special significance for the shell. They make the shell replace an expression with the list of files that match the expression. Because of this the expression needs to be "protected" from the shell. This is what the single quotes do. Consider the following:

```
$ ls
virgin  wizard

$ echo [Ww]izard
wizard

$ echo '[Ww]izard'
[Ww]izard
```

Exercise 7: Explain the following behaviour:

```
$ grep [Ww]izard wizard

$ grep '[Ww]izard' wizard
Speak, O Guru: How can I become a UNIX Wizard?
O, Nobly Born: know that the Way to Wizardhood is
Path of Wizardly Wisdom, first learn the
sole Path to Wizardry and the proper Way of every
Wizardhood?
Wizardhood shall surely come upon thee; but not
thyself in Pride claim the Mantle of Wizardry;
Wizards hail thee as a Brother or Sister in
```

This syntax is not restricted to pairs of (uppercase,lowercase) letters. Any set of characters can be inserted. For example, to find lines containing a lowercase "`c`", followed by a lowercase vowel, followed by a lowercase "`m`":

```
$ grep 'c[aeiou]m' wizard
Speak, O Guru: How can I become a UNIX Wizard?
the erratic but illuminating compendium called
by mere Study but must become One with it through
of those who came before thee.  Hack, and hack
Wizardhood shall surely come upon thee; but not
Rather must you Become, and Become, and Become,
```

It is also possible to specify a *range* of characters rather than a list. For example:

```
$ grep '[0-9]' virgin wizard
virgin:L610,000 in damages to Mr Richard Branson and his
virgin:From "The Independent", Tuesday 12th January, 1993.
virgin:[At a closed BA meeting in the late summer of 1990...]
virgin:Listening attentively was Sadig Khalifa, 41, a
wizard:_The_Design_Of_The_4.3BSD_UNIX_Operating System_
```

The expression "`[0-9]`" matches any character between "`0`" and "`9`" in the underlying character set. In the ASCII character set that Unix uses, this means all the digits so the expression "`[0-9]`" is exactly equivalent to "`[0123456789]`".

Exercise 8: Demonstrate that there are no pairs of adjacent digits in the file `wizard`.

If you ever need to match a set of characters that includes a hyphen, then make the hyphen the first character in the square brackets, where it cannot be interpreted as indicating a range, and it will be treated literally. So "`[-aeiou]`" will match any one lowercase vowel or a hyphen.

Exercise 9: How many lines are there in the file `virgin` with either a "`y`" or a "`-`" in them?

To combine ranges simply place them next to one another in the square brackets. So "`[A-Z0-9]`" will search for a single uppercase letter or digit:

```
$ grep -c '[A-Z0-9]' virgin
57
```

The expressions in the square brackets can be made more intricate than simple sets or ranges.

The expression "`[^p]`" means "any one character that is not a lowercase '`p`'". For example:

```
$ grep 'a[^p]ple' /usr/dict/words
ample
asplenium
counterexample
example
sample
trample
```

The file `/usr/dict/words` contains a list of words, one per line, and is commonly used for a spelling checker (American spelling).

Sets of characters can also be excluded in this way: "`[^aeiou]`" means any one character that is not a lowercase vowel and "`[^A-Z]`" means any one character that is not an uppercase letter:

```
$ grep 'a[^mp]ple' /usr/dict/words
asplenium

$ grep '[^aeiou][^aeiou][^aeiou][^aeiou][^aeiou][^aeiou]\
[^aeiou]'  /usr/dict/words
McIntyre
strychnine

$ grep -i '[^aeiou][^aeiou][^aeiou][^aeiou][^aeiou]\
[^aeiou][^aeiou]'  /usr/dict/words
strychnine
```

Exercise 10: How many lines in `/usr/dict/words` have six adjacent consonants? How many do if you count "`y`" as a vowel?

N.B. One particularly important set of characters is the set of "every character except the newline". The "newline" character is the character that Unix inserts between each line of text to mark their end. Rather than having the user type in a set of 125 characters within the square brackets the regular expression syntax gives "`.`" (full stop) this special meaning of "every character except the newline".

3.2 Selecting a number of characters

As we have just seen, the regular expression for "seven consonants in a row" is a bit cumbersome. There is a more convenient syntax available to specify "a certain number of characters from a certain set". Consider the following commands:

```
$ grep -i '[^aeiou][^aeiou][^aeiou][^aeiou][^aeiou]\
[^aeiou][^aeiou]'  /usr/dict/words
strychnine

$ grep -i '[^aeiou]\{7\}' /usr/dict/words
strychnine
```

Following any regular expression with a number within a "`\{...\}`" pair means that regular expression repeated that number of times.

Note that the *expression* is repeated, not the *match*. The regular expression "`[aeiou]\{4\}`" will match *any* four vowels in a row, not just four of the same vowels.

It is possible to specify ranges of numbers within the "`\{...\}`" pair. The expression "`[aeiou]\{3,5\}`" means "3, 4 or 5 vowels in a row" and the expression "`[aeiou]\{4,\}`" means "4 or more vowels in a row". For example:

```
$ grep -i 'qu[aeiou]\{3\}s' /usr/dict/words
aqueous
obsequious
```

will find all the words with "`qu`" followed by three vowels followed by "`s`". The next two examples find all those with 2 or 3 vowels between and all those with 2 or more.

```
$ grep -i 'qu[aeiou]\{2,3\}s' /usr/dict/words
acquiesce
acquiescent
aqueous
Iroquois
obsequious
queasy
quiescent
turquoise
```

```
$ grep -i 'qu[aeiou]\{2,\}s' /usr/dict/words
acquiesce
acquiescent
aqueous
Iroquois
obsequious
queasy
quiescent
turquoise
```

Exercise 11: Find all the words that have two or more vowels but no consonants in a row inside an "`s`...`s`" pair.

If you don't care how many characters matching the pattern there are then you could use "`\{0,\}`" meaning "zero or more". This is a bit cumbersome, though, so a shortcut is provided. An asterisk following an expression also means "zero or more". For example:

```
$ grep 'asc[a-z]*ism' /usr/dict/words
asceticism
fascism
```

Note that the second match ("`fascism`") has *no* characters between the "`asc`..." and the "...`ism`".

Exercise 12: How many words in `/usr/dict/words` have at least three instances of the letter "`y`" in them?

3.3 Defining the start and end of a line

A very useful facility in `grep` is to be able to define where the start and end of the line occur within the pattern being matched. For example, you might want "lines starting with '`aba`'" or "lines ending in '`ssing`'".

To specify the beginning of a line, use the character "`^`":

```
$ grep '^aba' /usr/dict/words
aback
abacus
abalone
abandon
abase
abash
abate
abater
```

and to specify the end of a line, use the character "$":

```
$ grep 'ssing$' /usr/dict/words
degassing
finessing
kissing
teleprocessing
```

The "^" and "$" characters are known as "anchors". They only have their special meaning at the start or end of a regular expression. Also, be careful not to confuse the use of "^" inside square brackets described in section 3.1 from its use outside described here.

Exercise 13: How many words in `/usr/dict/words` start with six consonants?

The two anchors can be combined. The regular expression "`^y[a-z]*y$`" will search for all lines that consist of a single word beginning and ending with a "y". As some of the words in the file begin with uppercase letters, a better expression might be "`^[Yy][a-z]*y$`". Alternatively the `-i` option could be used to make the entire search case insensitive.

Exercise 14: How many words in `/usr/dict/words` start with a "z" and end with an "a"?

3.4 Defining word boundaries

Just as "^" and "$" define the start and end of the line, there are regular expressions in `grep` that match the start and end of words. A word boundary is defined to be punctuation, spaces or either end of a line. The anchors for "start of word" and "end of word" are "\<" and "\>" respectively. Having these as separate anchors allows for more flexibility than the `-w` flag; it is possible to specify how a word should begin or end without having to specify how it should end or begin. For example, to find all the lines that have present participles ("`...ing`") in them we can give the command

```
$ grep 'ing\>' *
```

yielding 45 lines of output in total.

Exercise 15: How many lines in the file `virgin` have words begining with a capital vowel?

3.5 Recording previous matches

We have seen that the expression "`[aeiou][aeiou]`" will match a pair of vowels. We have, as yet, no syntax to specify a pair of the same vowels (*i.e.* "`aa`","`ee`","`ii`","`oo`", and "`uu`" but not "`ae`", "`ao`" *etc.*). This syntax will be provided now.

If part of a regular expression is surrounded by a "`\(...\)`" pair then that part of the expression will behave as if they were not there but the *result* of the match will be recorded for later use in the expression. For example:

```
$ grep '\([wxy]\)\1' /usr/dict/words
Exxon
hollowware
screwworm
```

Essentially, the "`\(...\)`" records matches in an internal store (called a "register" in the jargon) and the "`\1`" expression represents the contents of this register.

Exercise 16: How many words are there with double vowels (*i.e.* the *same* vowel twice) in `/usr/dict/words`?

Up to nine registers can be used. For example, "`\([a-z]\)\1\([a-z]\)\1\1\2`" will match anything of the pattern "*aabaab*":

```
$ grep '\([a-z]\)\1\([a-z]\)\1\1\2' /usr/dict/words
Mississippi
Mississippian
```

The first pair of "`\(...\)`" records "`s`" in both cases and stores it in "`\1`". The second pair records "`i`" in both cases and stores it in "`\2`".

It is not possible to nest these recording brackets.

3.6 Matching special characters

We have seen that certain characters are special within regular expressions. These are known in the jargon as "metacharacters". So what has to be done to search for a full stop ("`.`" being the metacharacter for "any character except the newline") or an open square bracket ("`[`" being the metacharacter starting a range of characters from which one will match) or the pair of characters "`\{`" (start of a numbering range) and so on?

With the exception of these characters that take special meaning when preceded by a backslash, any metacharacter can be made "ordinary" by preceding it with a backslash. So, to count the lines with full stops in them, we do the following:

```
$ grep -c '\.' *
virgin:33
wizard:41
```

Recall that the shell will expand "`*`" into every filename in the current directory. In this example it is equivalent to "`virgin wizard`". Leaving out the backslash would have tested for every line with at least one character on it:

```
$ grep -c '.' *
virgin:78
wizard:128
```

This prefixing of a backslash is called "escaping" the character. If you look for certain metacharacters on their own without escaping them then they are syntactically incorrect. For example:

```
$ grep '[' *
grep: RE error 49: [ ] imbalance.

$ grep '\[' *
virgin:Independent [a UK "quality" paper] describing the
virgin:[At a closed BA meeting in the late summer of 1990...]
virgin:When the BA team in the North Terminal [of
```

If the search is for the *pairs* of characters of which the first is a backslash and the second is one of the characters that is special after a backslash, then the backslash itself must be escaped. This involves preceding the backslash with another backslash:

```
$ grep -c '\(' *
grep: RE error 42: \( \) imbalance.

$ grep -c '\\(' *
virgin:0
wizard:0
```

To see an example that has a matching line in it, and demonstrates that the single quotes are needed, try the following:

```
$ echo \(
(

$ echo '\('
\(

$ echo '\(' | grep '\('
grep: RE error 42: \( \) imbalance.

$ echo '\(' | grep '\\('
\(
```

3.7 The `egrep` command

The `egrep` command has facilities for regular expressions that are missing some of `grep`'s features but possesses one useful addition:

There are no facilities for word delimiters. `egrep` lacks the "`-w`" option and does not have the "`\<`" and "`\>`" word anchors.

There are no facilities for recording and reusing matches with "`\(`", "`\)`" and "`\1`" *etc.*

There are no facilities for counting repetitions with "`\{...\}`".

`egrep` *does* provide facilites for alternation; it is possible to search with `egrep`, but not in `grep`, for "either this expression *or* this expression". To do this, the expressions should be separated by single vertical bar:

```
$ egrep 'computer|BABS' virgin
cracking of a computer by BA staff to get
computer systems.
tapping into our computer terminals in the
out a segment of a vast computer known as BABS
could see on the BABS computer system when flight
Gatwick airport] learnt from the Virgin computer
information they were hacking out of BABS. The
so were their computer terminals. While one of
to get into the computer. We had no reason for
computer but failed. Ogden had issued their staff
travel timetables directly from BABS. This
```

`a`	The letter “a”
`[abc]`	Any one of “a”, “b” or “c”.
`[a-e]`	Any one of “a”, “b”, “c”, “d” or “e”.
`[a-cA-C]`	Any one of “a”, “b”, “c”, “A”, “B” or “C”.
`[^abc]`	Any one character except “a”, “b” or “c”.
`[aeiou]\{3\}`	Three lowercase vowels in a row.
`[aeiou]\{3,\}`	Three or more lowercase vowels in a row.
`[aeiou]\{3,6\}`	Three, four, five or six lowercase vowels in a row.
`[aeiou]\{,3\}`	Zero, one, two or three lowercase vowels in a row.
`[aeiou]*`	Zero or more lowercase vowels.
`.`	Any single character except the newline
`^`	Start of a line (at the end of a reg. exp.)
`$`	End of a line (at the end of a reg. exp.)
`\<`	Start of a word
`\>`	End of a word
`\(...\)`	Records the match of the enclosed reg. exp. in `\1`, `\2` *etc.*
`\1`	The result of the reg. exp. in the first `\(...\)` pair
`\.`	A full stop
`\[`	A square bracket

Table 3.1: Summary of `grep` regular expression metacharacters