Review: Unix Commands

- How can we execute a command within a command?
- What should we use if we have too may files to pass in as arguments to a command?
- What are regular expressions?
  - What can we represent with regular expressions?
  - How do we represent those things?

Today

- Regular Expressions
- grep

Regular Expressions

- The simplest regular expressions are a string of literal characters to match
- The string ***matches*** the regular expression if it contains the substring

Review: Regular Expressions

- In the command line, * means any character
- In other contexts,
  - * means 0 or more of the previous character(set)
  - . means any character

Repetition Ranges

- Ranges can also be specified
  - `{ }` notation can specify a range of repetitions for the immediately preceding regex
  - `{n}` means exactly *n* occurrences
  - `{n,}` means at least *n* occurrences
  - `{n,m}` means at least *n* occurrences but no more than *m* occurrences
- Examples:
  - `.{0,}` same as `.*`
  - `a{2,}` same as `aaa*`

Subexpressions

- If you want to group part of an expression so that `*` or `{ }` applies to more than just the previous character, use `( )` notation
- Subexpresssions are treated like a single character
  - `a*` matches 0 or more occurrences of `a`
  - `abc*` matches `ab`, `abc`, `abcc`, `abccc`, …
  - `(abc)*` matches `abc`, `abcabc`, `abcabcabc`, …
  - `(abc){2,3}` matches `abcabc` or `abcabcabc`

grep Family

- **g**lobally search a **re**gular expression and **p**rint

- `grep` - uses regular expressions for pattern matching
- `fgrep` - file grep
  - does not use regular expressions, only matches fixed strings (considered as literals) but can get search strings from a file
  - generally the *fastest* member of the grep family
- `egrep` - extended grep
  - uses a more powerful set of regular expressions
  - does not support backreferencing

Syntax

- Regular expression concepts we have seen so far are common to grep and egrep
- `grep` and `egrep` have slightly different syntax
  - `grep`: Basic Regular Expressions BREs
  - `egrep`: EREs (enhanced features we will discuss)
- Major syntax differences:
  - `grep`: \(and \), \{ and \}
  - `egrep`: (and), { and }

Protecting Regex Metacharacters

- Many special characters used in regexs also have special meaning to the shell

Quote your regexs

  - Protects special characters from being operated on by the shell
  - If you habitually do it, you won't have to worry about when it is necessary

Escaping Special Characters

- To get literal characters, escape the character with a \ (backslash)
- Suppose we want to search for the character sequence a*b*
  - a*b* will match zero or more 'a's followed by zero or more 'b's (not what we want)
  - Use a*b*
    - Asterisks are now treated as regular characters

Egrep: Alternation

- Regex also provides an alternation character | for matching one or another subexpression
  - (T|Fl)an will match 'Tan' or 'Flan'
  - ^(From|Subject): will match the From and Subject lines of a typical email message
    - It matches a beginning of line followed by either the characters 'From' or 'Subject' followed by a ':'
- Subexpressions are used to limit the scope of the alternation
  - At(ten|nine)tion then matches "Attention" or "Atninetion"
  - Atten|ninetion would match "Atten" or "ninetion"

Egrep: Repetition Shorthands

- * (star) specifies zero or more occurrences of the immediately preceding character
- + (plus) means "one or more"
  - abc+d will match 'abcd', 'abccd', or 'abccccccd' but will not match 'abd'
  - Equivalent to {1,}
- ? (question mark) specifies an *optional* character
  - Single character that immediately precedes it
  - July? will match 'Jul' or 'July'
  - Equivalent to {0,1} and (Jul|July)

Egrep: Repetition Shorthands

- *, ?, and + are known as ***quantifiers*** because they specify the *quantity* of a match
- Quantifiers can also be used with subexpressions
  - `(a*c)+`

Egrep: Repetition Shorthands

- *, ?, and + are known as ***quantifiers*** because they specify the *quantity* of a match
- Quantifiers can also be used with subexpressions
  - `(a*c)+` matches 'c', 'ac', 'aac' or 'aacaacac' but will not match 'a' or a blank line

Practical Regex Examples

- Variable names in C/Python
- Dollar amount with optional cents
- Time of day
- HTML headers <h1> <H1> <h2> …

Make some test cases and try out your expressions

Practical Regex Examples

- Variable names in C/Python
  - `[a-zA-Z_][a-zA-Z_0-9]*`
- Dollar amount with optional cents
  - `\$[0-9]+(\.[0-9][0-9])?`
- Time of day
  - `(1[012]|[1-9]):[0-5][0-9] (am|pm)`
- HTML headers <h1> <H1> <h2> …
  - `<[hH][1-6]>`
  - New standard is lower case h

Grep: Backreferences

- ***Backreferences*** allow us to refer to a match that was made earlier in a regex
  - `\n` is the backreference specifier, where n is a number
  - Looks for nth subexpression
- Example: HTML Tags
  - `<h[1-6]>.*</h[1-6]>` is not good enough to match html headers, since it matches `<h1>Hello world</h3>`
  - `<h\([1-6]\)>.*</h\1>` matches what we were trying to match before.

Grep: Backreference Examples

- To find if the first word of a line is the same as the last:
  - `^\([[:alpha:]]\{1,\}\) .* \1$`
  - `\([[:alpha:]]\{1,\}\)` matches 1 or more letters
- Another example:
  - `"Mr \(dog\|cat\) came home to Mrs \1 and they went to visit Mr \(dog\|cat\) and Mrs \2 to discuss the meaning of life"`

What text should this match?

grep Family Syntax

grep [-hilnv] [-e expression] [filename]
egrep [-hilnv] [-e expression] [-f filename] [expression] [filename]
fgrep [-hilnxv] [-e string] [-f filename] [string] [filename]

Option	Meaning
`-h`	Do not display filenames
`-i`	Ignore case
`-l`	List only filenames containing matching lines
`-n`	Precede matching line with its line number
`-v`	Select non-matching lines
`-x`	Match whole line only
`-e expression`	Specify expression as option
`-f filename`	Take regular expression (egrep) or a list of strings (fgrep) from filename

Fun with the Dictionary

- **`/usr/share/dict/words`** contains over 400,000 words
  - **`egrep hh /usr/share/dict/words`**
    - aarrghh
    - Ahhiyawa
    - archhead
    - archheart

    …
- **egrep** as a simple spelling checker: Specify plausible alternatives you know

```
egrep "n(ie|ei)ther" /usr/share/dict/words
Neither
```

- How many words have 3 a's one letter apart? 3 u's?

Fun with the Dictionary

- How many words have 3 a's one letter apart?
  - `egrep a.a.a /usr/share/dict/words | wc -l`
    - 1632
- How many words have 3 u's one letter apart?
  - `egrep u.u.u /usr/share/dict/words | wc -l`
    - 84

grep Examples

- `grep 'men' greptest`
- `grep 'fo*' greptest`
- `egrep 'fo+' greptest`
- `egrep -n '[Tt]he' greptest`
- `fgrep 'The' greptest`