The ‘grep’ Command

Colin Masterson

PDF created from
https://www.cs.drexel.edu/~knowak/cs265_spring_2011_d/CM_The_grep_Command.ppt

What is grep?

- grep is a command that searches through the input file for a specified pattern
- When grep finds a match to the pattern, it prints the entire line to standard output
- grep general structure:
  - grep options pattern input_file_names

Options

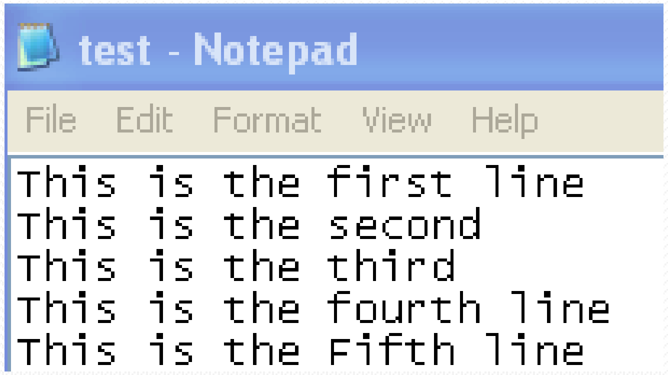
- grep has a variety of options that can execute a wide range of operations once a match is found
- This presentation is an overview of some of the more basic/common options
- For all available options and more in depth explanations, please see Unix resource 3 on the course website titled “Grep command documentation” or check the man page for grep in your terminal

Matching Control

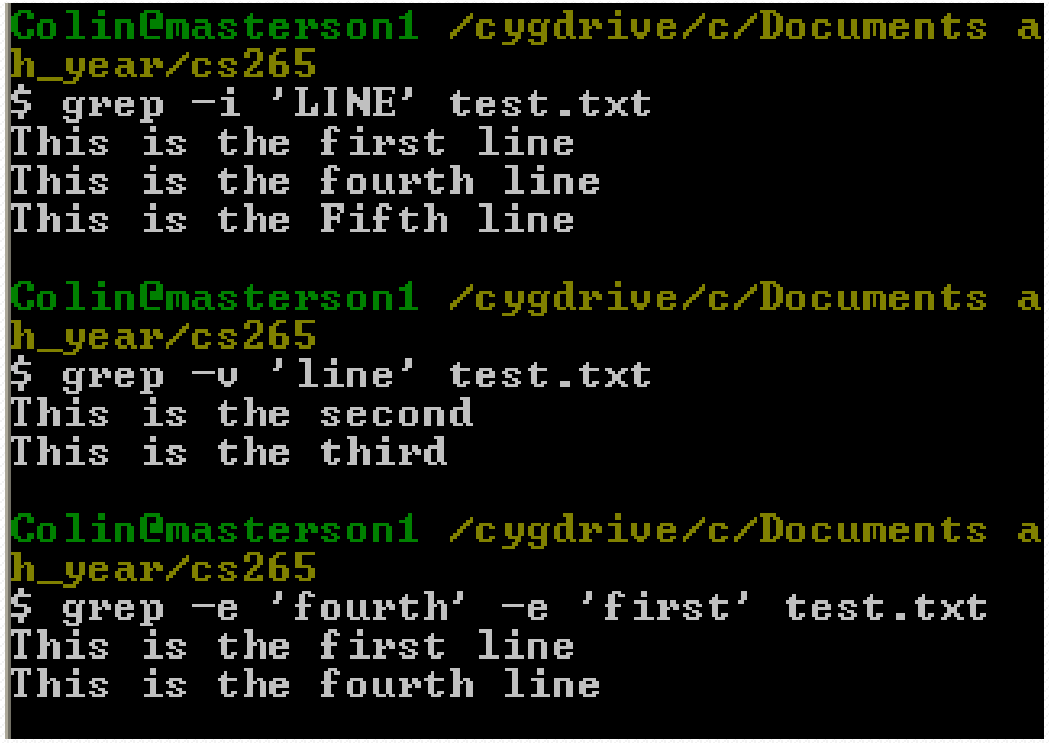
- ‘-i’
  - Ignores case.
- ‘-v’
  - Inverts the matching. When used, grep will print out lines that do not match the pattern
- ‘-e pattern’
  - Pattern is the pattern. This can be used to specify multiple patterns, or if the pattern starts with a ‘-’. A line only has to contain one of the patterns to be matched.

Examples using ‘-i’, ’-v’, and ‘-e’

- The file below was used for these examples



```
This is the first line
This is the second
This is the third
This is the fourth line
This is the Fifth line
```



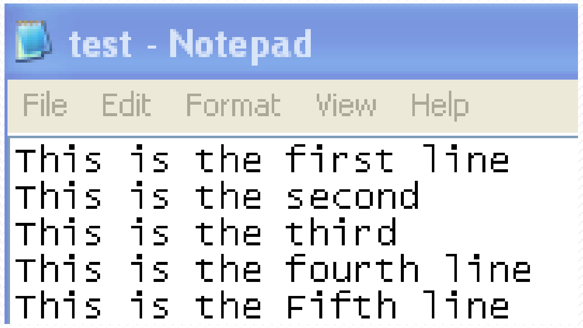
```
Colin@masterson1 /cygdrive/c/Documents a
h_year/cs265
$ grep -i 'LINE' test.txt
This is the first line
This is the fourth line
This is the Fifth line

Colin@masterson1 /cygdrive/c/Documents a
h_year/cs265
$ grep -v 'line' test.txt
This is the second
This is the third

Colin@masterson1 /cygdrive/c/Documents a
h_year/cs265
$ grep -e 'fourth' -e 'first' test.txt
This is the first line
This is the fourth line
```

General Output Control

- ‘-c’
  - Suppress normal output and instead print out a count of matching lines for each input file
- ‘-l’
  - Suppress normal output and print the name of each file that contains at least one match
- ‘-L’
  - Suppress normal output. Print the name of each file that does not contain any matches
- Note: both the ‘-l’ and ‘-L’ options will stop searching a file once a match is found

Examples using '-c', '-l', and '-L'

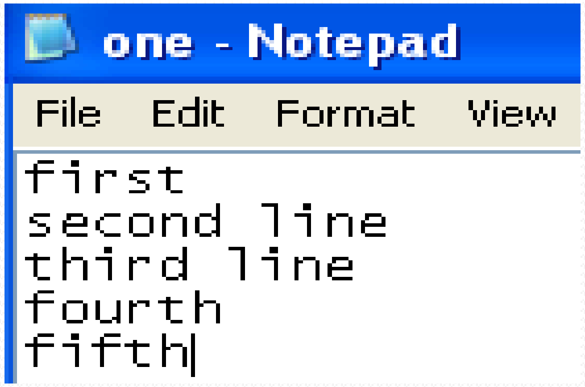
- Two files were used and specified below



test - Notepad

File Edit Format View Help

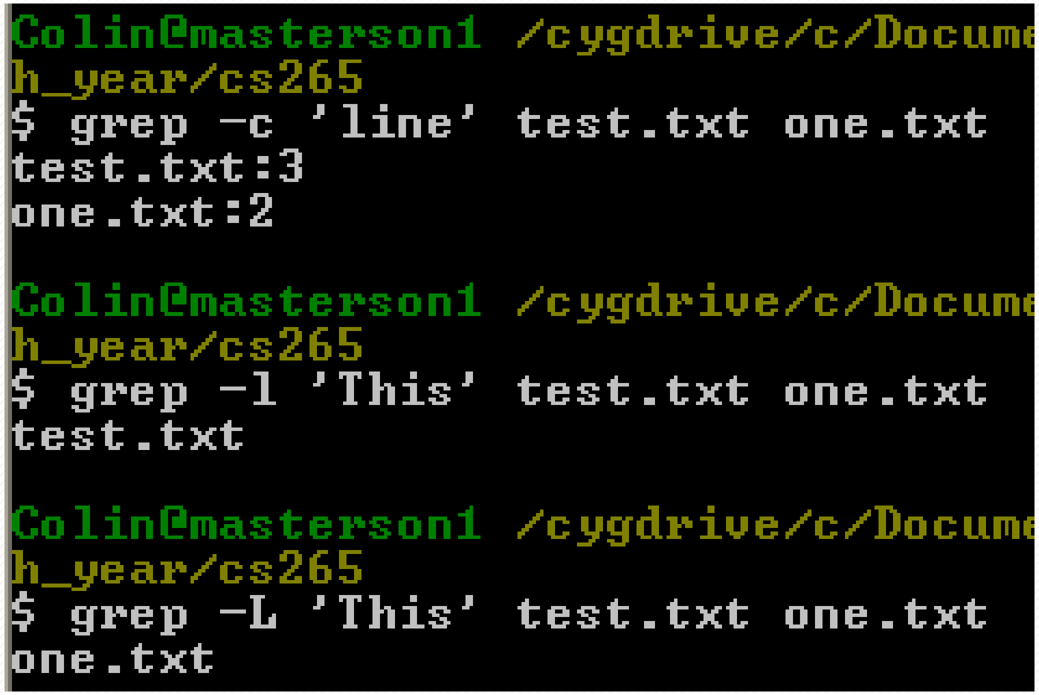
```
This is the first line
This is the second
This is the third
This is the fourth line
This is the Fifth line
```



one - Notepad

File Edit Format View

```
first
second line
third line
fourth
fifth
```



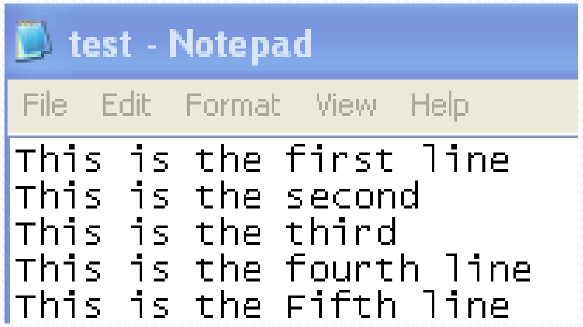
```
Colin@masterson1 /cygdrive/c/Docume
h_year/cs265
$ grep -c 'line' test.txt one.txt
test.txt:3
one.txt:2

Colin@masterson1 /cygdrive/c/Docume
h_year/cs265
$ grep -l 'This' test.txt one.txt
test.txt

Colin@masterson1 /cygdrive/c/Docume
h_year/cs265
$ grep -L 'This' test.txt one.txt
one.txt
```

Output Line Prefix Control

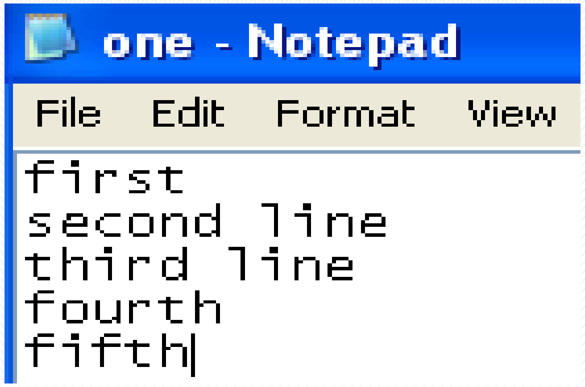
- ‘-n’
  - Prefixes each line of output with the line number from the input file the match was found on
- ‘-H’
  - Prefix each line of output with the input file name that the match was found in
- ‘-T’
  - Makes sure that the actual line content (or whatever content comes after the ‘-T’) lands on a tab stop

Examples using ‘-H’, ’-n’, and ‘-T’

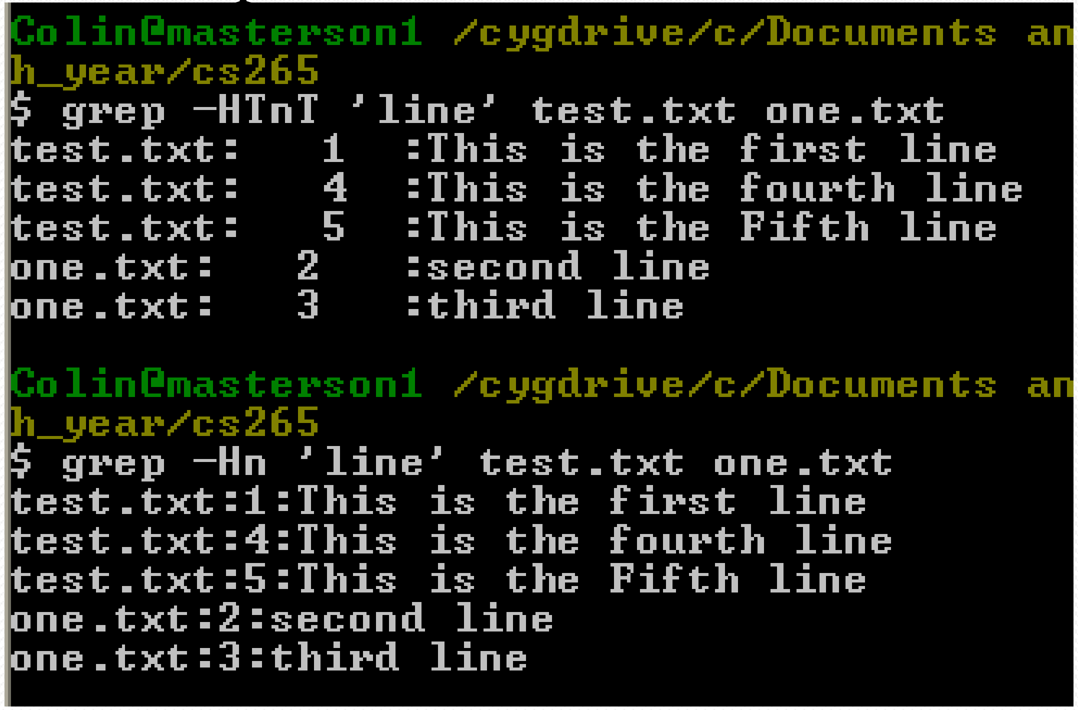
- Two files were used and specified below



```
This is the first line
This is the second
This is the third
This is the fourth line
This is the Fifth line
```



```
first
second line
third line
fourth
fifth
```



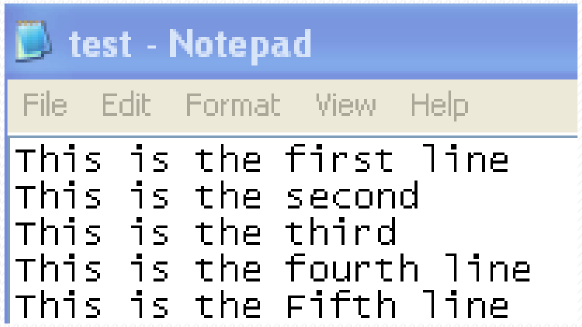
```
Colin@masterson1 /cygdrive/c/Documents an
h_year/cs265
$ grep -HTnT 'line' test.txt one.txt
test.txt:    1   :This is the first line
test.txt:    4   :This is the fourth line
test.txt:    5   :This is the Fifth line
one.txt:    2    :second line
one.txt:    3    :third line

Colin@masterson1 /cygdrive/c/Documents an
h_year/cs265
$ grep -Hn 'line' test.txt one.txt
test.txt:1:This is the first line
test.txt:4:This is the fourth line
test.txt:5:This is the Fifth line
one.txt:2:second line
one.txt:3:third line
```

Context Line Control

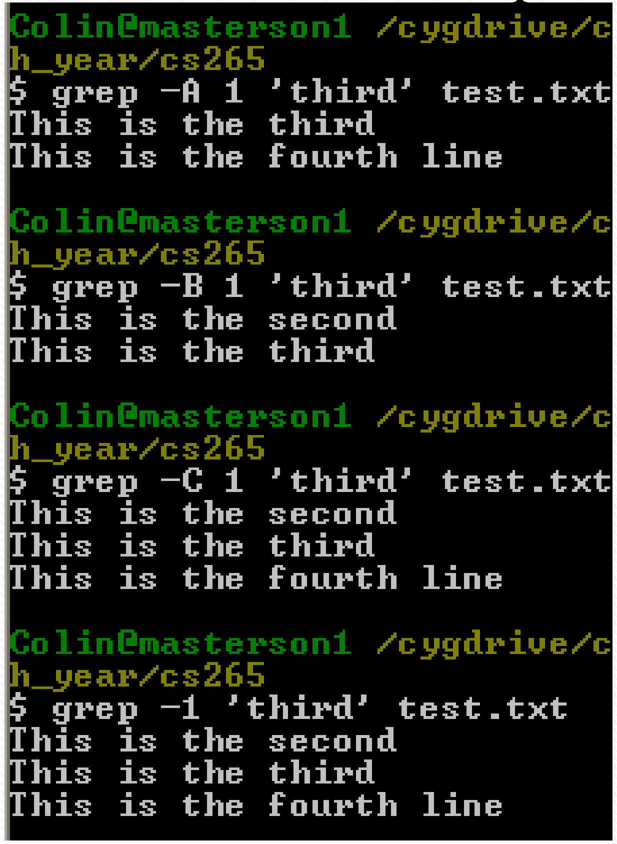
- '-A *num*'
  - Print *num* lines of trailing context after matching lines
- '-B *num*'
  - Print *num* lines of leading context before matching lines
- '-C *num*' or '-*num*'
  - Print *num* lines of leading and trailing output context

Examples using '-A', '-B', and '-C'

- The file below was used for these examples



```
This is the first line
This is the second
This is the third
This is the fourth line
This is the fifth line
```

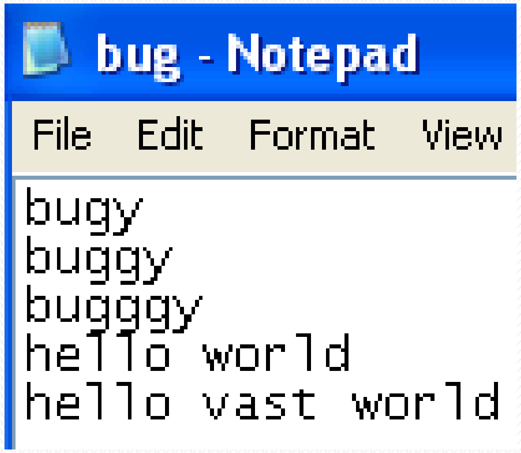


```
Colin@masterson1 /cygdrive/c
h_year/cs265
$ grep -A 1 'third' test.txt
This is the third
This is the fourth line

Colin@masterson1 /cygdrive/c
h_year/cs265
$ grep -B 1 'third' test.txt
This is the second
This is the third

Colin@masterson1 /cygdrive/c
h_year/cs265
$ grep -C 1 'third' test.txt
This is the second
This is the third
This is the fourth line

Colin@masterson1 /cygdrive/c
h_year/cs265
$ grep -1 'third' test.txt
This is the second
This is the third
This is the fourth line
```

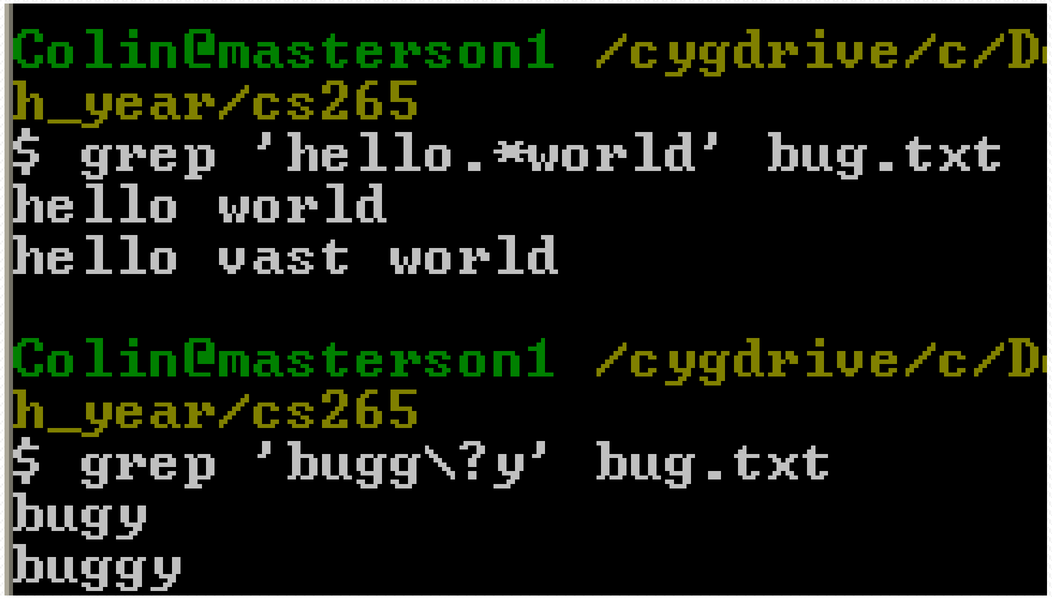
Special Characters

- '.' The period '.' matches any single character.
- '?' The preceding item is optional and will be matched at most once.
- '*' The preceding item will be matched zero or more times.
- '+' The preceding item will be matched one or more times.
- '{n}' The preceding item is matched exactly n times.
- '{n,}' The preceding item is matched n or more times.
- '{,m}' The preceding item is matched at most m times.
- '{n,m}' The preceding item is matched at least n times, but not more than m times.

Examples using '.', '*', and '?'



```
bugy
buggy
bugggy
hello world
hello vast world
```

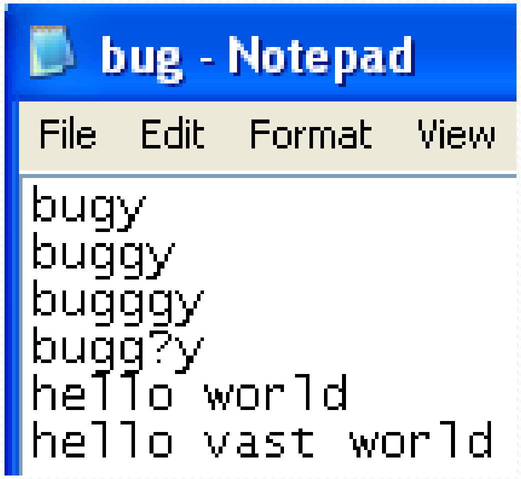


```
Colin@masterson1 /cygdrive/c/Do
h_year/cs265
$ grep 'hello.*world' bug.txt
hello world
hello vast world

Colin@masterson1 /cygdrive/c/Do
h_year/cs265
$ grep 'bugg\?y' bug.txt
bugy
buggy
```

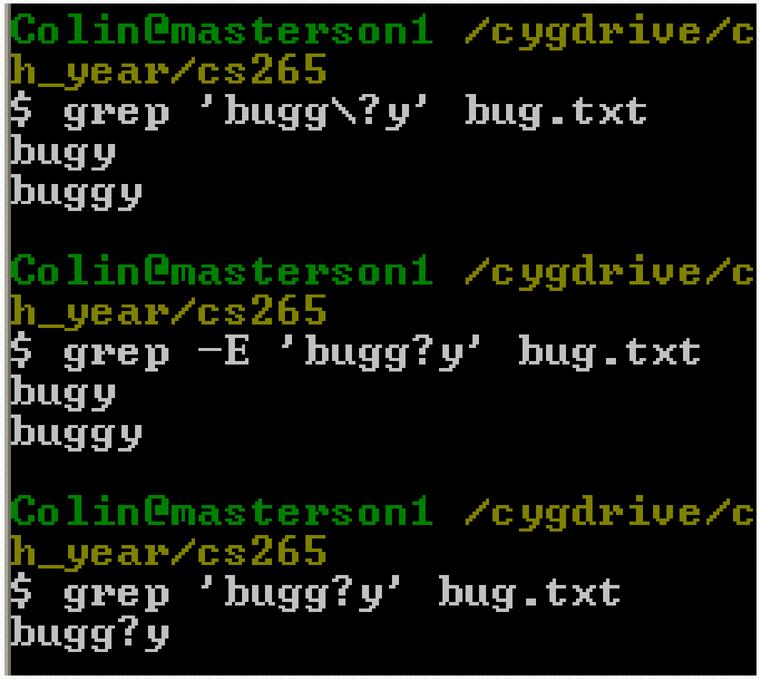
Basic vs Extended Regular Expressions

- ‘-G’
  - Interpret pattern as basic regular expression (BRE). This is the default.
- ‘-E’
  - Interpret pattern as extended regular expression (ERE)
- When using basic regular expression some special characters (like ‘?’ in the previous example) loose their special meaning and must have a ‘\’, the escape character, before them
- When using ERE, the escape character is unnecessary

BRE and ERE Difference



```
bugy
buggy
bugggy
bugg?y
hello world
hello vast world
```



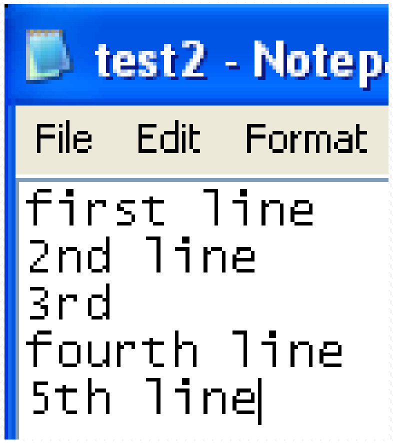
```
Colin@masterson1 /cygdrive/c
h_year/cs265
$ grep 'bugg\?y' bug.txt
bugy
buggy

Colin@masterson1 /cygdrive/c
h_year/cs265
$ grep -E 'bugg?y' bug.txt
bugy
buggy

Colin@masterson1 /cygdrive/c
h_year/cs265
$ grep 'bugg?y' bug.txt
bugg?y
```

- Note that without the '\' in the BRE call (example 3), the '?' is seen as a normal character

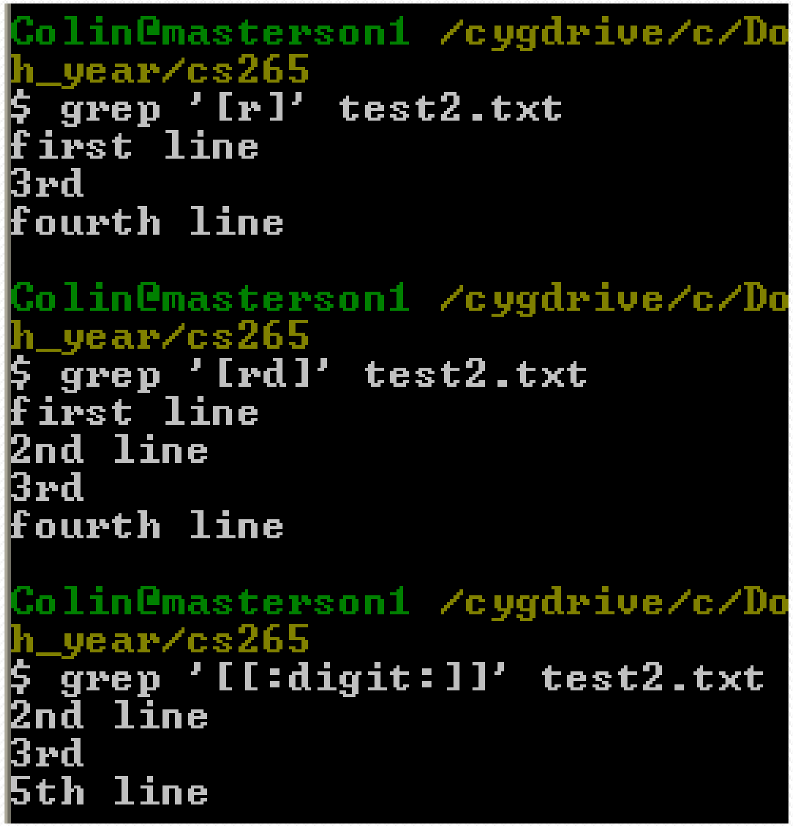
Bracket Expressions

- A bracket expression is a list of characters enclosed by '[' and ']'. It matches any single character in the list
- However, if the first character in the list is '^', it matches any character not in the list
- A range can be done by using '-' in a bracket expression
  - [0-5] is the same as [012345]
- Some ranges are pre-defined in character classes
  - [:digit:] is the same as 0123456789
  - When using grep, the class name (including brackets) must be contained within another set of brackets

Bracket Expression Example



```
first line
2nd line
3rd
fourth line
5th line
```



```
Colin@masterson1 /cygdrive/c/Do
h_year/cs265
$ grep '[r]' test2.txt
first line
3rd
fourth line

Colin@masterson1 /cygdrive/c/Do
h_year/cs265
$ grep '[rd]' test2.txt
first line
2nd line
3rd
fourth line

Colin@masterson1 /cygdrive/c/Do
h_year/cs265
$ grep '[[:digit:]]' test2.txt
2nd line
3rd
5th line
```