Advanced utilities

Introduces utilities for power users, grouped into logical sets

We introduce about thirty useful utilities.

Section	Utilities
Filtering files	egrep, fgrep, grep, uniq
Sorting files	sort
Extracting fields	cut
Comparing files	cmp, diff
Archiving files	tar, cpio, dump
Searching for files	find
Scheduling commands	at, cron, crontab
Programmable text processing	awk, perl
Hard and soft links	ln
Switching users	su
Checking for mail	biff
Transforming files	compress, crypt, gunzip, gzip, sed, tr, ul, uncompress
Looking at raw file contents	od
Mounting file systems	mount, umount
Identifying shells	whoami
Document preparation	nroff, spell, style, troff
Timing execution of commands	time

uniq

- **uniq** is the tool that helps to detect the adjacent duplicate lines and also deletes the duplicate lines

```
//...syntax of uniq...//
$uniq [OPTION] [INPUT[OUTPUT]]
```

```
$cat kt.txt
I love music.
I love music.
I love music.
```

```
$uniq kt.txt
I love music.
```

YORK UNIVERSITÉ UNIVERSITY

Filtering Files grep

- **grep, egrep, fgrep** "Global/Get Regular Expression and Print"

  -w -i -v

  - Filter out all lines that do not contain a specified pattern,
  - Giving you the line that contains the specified pattern

- **uniq**: which filters out duplicate adjacent lines.
  - **-i**: Prints lines with matching criteria while ignores casing (Upper/Lowecase).
  - **-l**: Prints filenames only.
  - **-n**: Prints lines with matching criteria and line numbers.
  - **-c**: Prints count of lines with matching criteria.
  - **-v**: Prints lines not matching criteria (inverse search).
  - **-w**: Prints whole word matches.
  - **-A n**: Prints *n* lines after matches.
  - **-B n**: Prints *n* lines before matches.
  - **-C n**: Prints *n* lines before and after matches.

Filtering Files grep

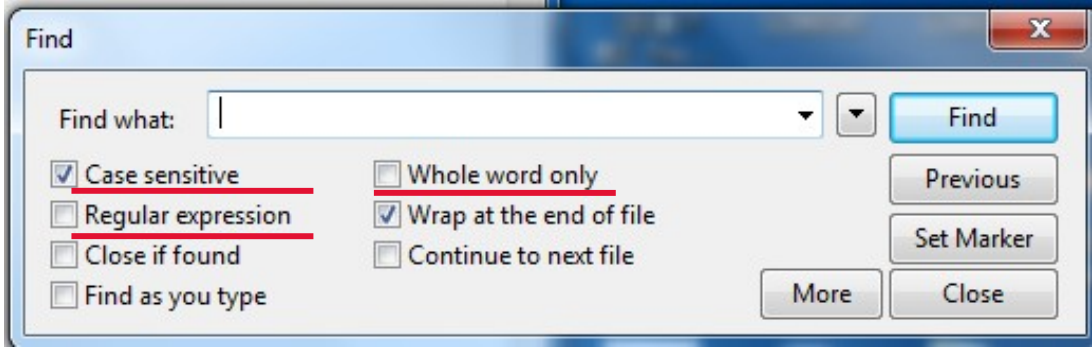
- **grep, egrep, fgrep** "Global/Get Regular Expression and Print"

  -w -i -v

  - Filter out all lines that do not contain a specified pattern,
  - Giving you the line that contains the specified pattern

```
$ cat inputFile .txt      # list the file to be filtered
line1 Well you know it's your bedtime,
line2 So turn off the light,
line3 Say all your prayers and then,
line4 Oh you sleepy young heads dream of wonderful things,
line5 Beautiful mermaids will swim through the sea,
line6 And you will be swimming there too.
$ grep the inputFile.txt      # search for the word "the"
line2 So turn off the light,
line3 Say all your prayers and then,
line5 Beautiful mermaids will swim through the sea,
```

Searching for Regex: grep



$ grep -w the inputFile.txt # -w: Whole word only

Lines that contain “the” as whole word

line2 So turn off the light,

line5 Beautiful mermaids will swim through the sea,

$ cat inputFile.txt | grep -w the

$ grep -v -w the inputFile.txt # -v: reverse the filter.

Lines that don’t contain “the” as whole word

line1 Well you know it’s your bedtime,

line3 Say all your prayers and then,

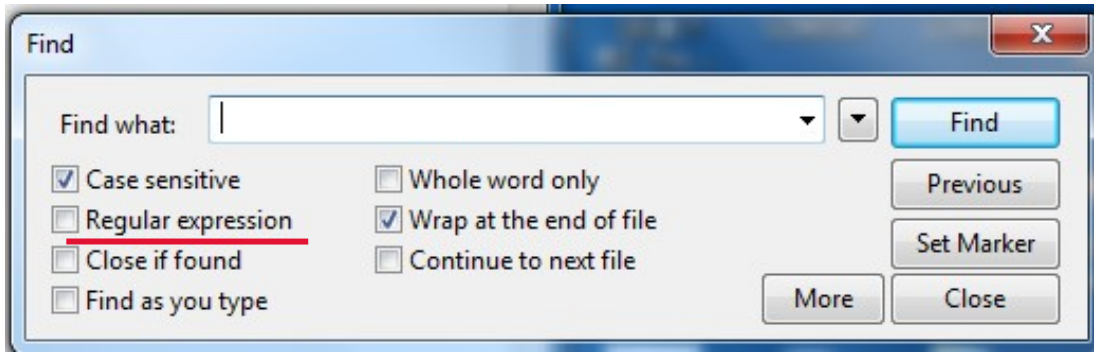
line4 Oh you sleepy young heads dream of wonderful things,

line6 And you will be swimming there too.

$ grep -i -w the inputFile.txt # ignore case, default case sensitive

YORK UNIVERSITÉ UNIVERSITY

Searching for Regex: grep



How to use grep to search lines that contain numbers?

$ grep ? inputFile.txt

How to use grep to search lines that contain lower case letters?

$ grep ? inputFile.txt

Utility	Kind of pattern that may be searched for
fgrep	fixed string only
grep	regular expression
egrep	extended regular expression

Regular Expressions

What is a Regular Expression?

- A regular expression (regex) describes a pattern to match multiple input strings.

- Regular expressions descend from a fundamental concept in Computer Science called finite automata theory

- Regular expressions are ubiquitous to Unix
  - Some utilities/programs that use Regex:
    - **vi**, **ed**, **sed**, and **emacs**
    - **awk**, **tcl**, **perl** and **Python**
    - **grep**, **egrep**
    - **Libraries** **scanf (" %[^\n]s ", str);**

For this course

- The simplest regular expression is a string of literal characters to match.

- The string **matches** the regular expression if it contains the substring.

Regular Expressions

Exact Matches

Match one any char .

Alternate []

[ab]
[^ab]
[a-d]

Anchors ^ $

Repetitions

* 0 or more

? 0 or 1

\+ 1 or more

YORK UNIVERSITÉ UNIVERSITY

Regular Expressions: Exact Matches

regular expression → cks

```
$ grep  cks  inputFile.txt
```

UNIX Tools rocks.

match

UNIX Tools sucks.

match

UNIX Tools is okay.

no match

YORK UNIVERSITÉ UNIVERSITY

Regular Expressions: Multiple Matches

- A regular expression can match a string in more than one place.

regular expression → apple

$ grep apple inputFile.txt

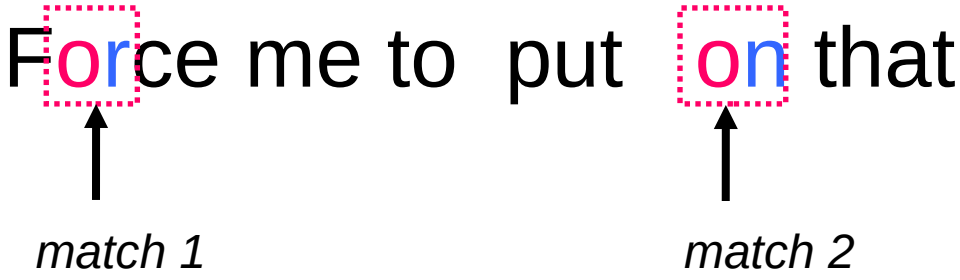
$ grep -w apple inputFile.txt ?

Regular Expressions: Matching Any Character

- The . regular expression can be used to match any one character.

regular expression → o.

$ grep o. inputFile.txt



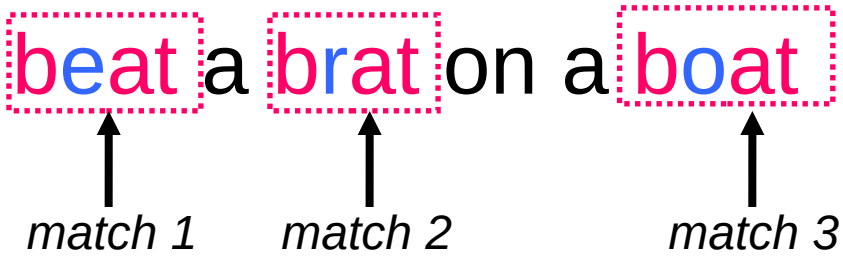
$ grep -w o. inputFile.txt ?

Regular Expressions: Alternate Character Classes

- Character classes **[ ]** can be used to match any specific set of characters.

regular expression ⟶ b [eor] a t

$ grep b[eor]at inputFile.txt



Does not match beoat
b[eor][or]at will do bat?
Does not match bat

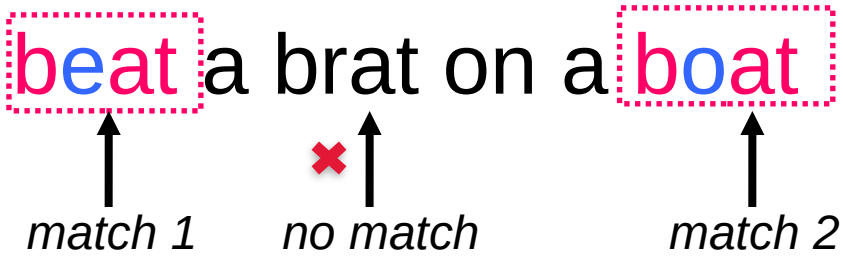
- [aeiou] will match any of the characters a, e, i, o, u
- [kK]orn will match korn or Korn

Regular Expressions: Alternate Character Classes

- Character classes [ ] can be used to match any specific set of characters.

regular expression → b [eo] a t

$ grep b[eo]at inputFile.txt



Does not match beoat
b[eor][or]at will do bat?
Does not match bat

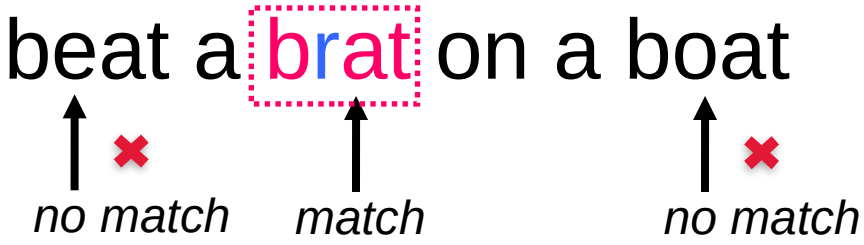
- [aeiou] will match any of the characters a, e, i, o, u
- [kK]orn will match korn or Korn

Regular Expressions: Negated Character Classes

- Character classes can be negated with the **[^]** syntax.
  - Negate all in []

regular expression → b [^eo] a t

$ grep b[^eo]at inputFile.txt



Regular Expressions: Other Character Classes

- Other examples of character classes:
  - [0123456789] will match any digit
  - [abcde] will match a b c d e

- Ranges can also be specified in character classes

  [0-9] is the same as [0123456789]
  [a-e] is equivalent to [abcde]

  $ grep [0-9] inputFile.txt

- You can also combine multiple ranges

  [abcde123456789] is equivalent to [a-e1-9]

[aeiou] 

Regular Expressions: Named Character Classes

- Commonly used character classes can be referred to by name
  - alpha,
  - lower,
  - upper,
  - alnum,
  - digit,
  - Punct (all non-word and non-space characters.),

For your information

- Syntax [[:name:]]
  - [0-9] [[:digit:]] $ grep [[:digit:]] inputFile
  - [a-zA-Z] [[:alpha:]]
  - [a-zA-Z0-9] [[:alnum:]]
  - [45a-z] [45[:lower:]]

- Important for portability across languages

YORK UNIVERSITÉ UNIVERSITY

Exact Matches

Match one any char .

Alternate []

- [ab]
- [^ab]
- [a-d]

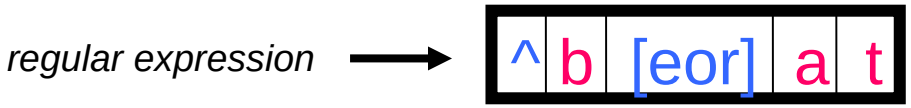
Anchors ^ $

Repetitions

- * 0 or more
- ? 0 or 1
- + 1 or more

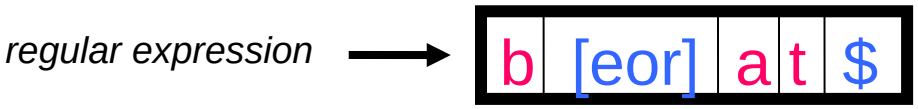
Regular Expressions: Anchors

- Anchors are used to match at the beginning or end of a line (or both).
  - ^ means beginning of the line ^ the begin with “the”
  - $ means end of the line the$ end with “the”



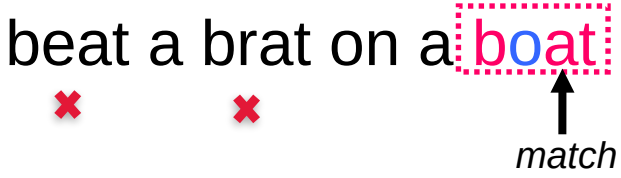
$ grep ^b[eor]at inputFile.txt

This is brat on a boat ?



$ grep b[eor]at$ inputFile.txt

A brat on a boat, rght ?



$

Middle

YORK UNIVERSITÉ UNIVERSITY

Regular Expressions: Anchors

- Anchors are used to match at the beginning or end of a line (or both).
  - ^ means beginning of the line ^ the begin with “the”
  - $ means end of the line the$ end with “the”

$grep cse classlist

```
red 32 % grep cse classlist
cse***          *********       Yu      Ying
cse***          *********       Wong    JunXiu
ttCTV*          cse******       Tong    Tracy
red 33 %
```

$grep ^cse classlist

```
red 33 % grep ^cse classlist
cse***          *********       Yu      Ying
cse***          *********       Wong    JunXiu
red 34 %
```

Exact Matches

Match one any char .

Alternate []

[ab]
[^ab]
[a-d]

Anchors ^ $

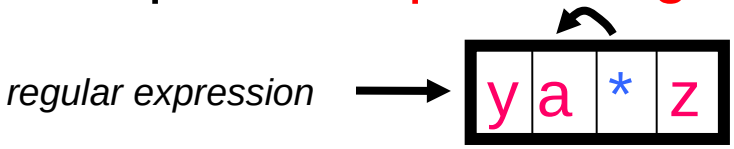
Repetitions

* 0 or more
? 0 or 1
+ 1 or more

Regular Expression: Repetitions

"Kleene Star"

- The * is used to define **zero or more** occurrences of the *single* regular expression preceding it.



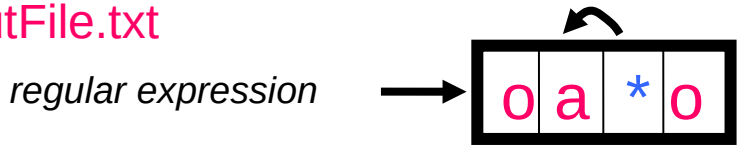
$ grep ya*z inputFile.txt

I got mail, yaaaaaaaaaaz!

match

zero or more occurrences of 'a' (between y z)
yz yaz yaaz yaaaz

$ grep oa*o inputFile.txt



For me to look on. Take oaao

match *match*

zero or more occurrences of 'a' (between o o)
oo oao oaao oaaao

Regular Expressions: Repetition Ranges, Subexpressions

- Ranges can also be specified
  - `{n,m}` 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
- Example:

  `.{0,}` same as `.*`

  `a{2,}` same as `aaa*` **# at least 2 occurrences**

  `a{2}` same as `aa` **# exact 2 occurrences**

For your information

- If you want to group part of an expression so that `*` applies to more than just the previous character, use `( )` notation

- Subexpresssions are treated like a single character

  `a*` matches zero or more occurrences of `a`

  `abc*` matches `ab`, `abc`, `abcc`, `abccc`, … **# ab followed by 0 or more c**

  `a(bc)*` matches `a`, `abc`, `abcbc`, `abcbcbc`, …

  `(abc)*` matches `abc`, `abcabc`, `abcabcabc`, …

Extended Regular Expressions: Repetition Shorthands

- The * (star) has already been seen to specify zero or more occurrences of the immediately preceding character
  - **abc*d** will match abd, abcd, abccd, or abccccccd

- The + (plus) means one or more occurrence of the preceding character
  - **abc+d** will match abcd, abccd, or abccccccd
    but will not match abd ✘
    one or more occurrence of c

- The ? (question mark) specifies an optional character, the single character that immediately precedes it
  - **July?** will match Jul or July
    - Equivalent to **(Jul|July)**
    zero or one occurrence of y
  - **abc?d** will match abd and abcd
    but will not match abccd ✘

Repetition * ? + summary

Regex	Meaning
a*	0 or more a
a?	0 or 1 a
a+	1 or more a

ab*c matches

ab?c matches

ab+c matches

Which infinite or finite?

Repetition * ? + summary

Regex	Meaning
a*	0 or more a
a?	0 or 1 a
a+	1 or more a

ab*c matches ac abc abbc abbbc abbbbc ….

ab?c matches ac abc

ab+c matches abc abbc abbbc abbbbc ….

- Don’t get confused with filename wildcard *

  ls ba* ba followed by 0 or more any char -- anything

  ls a*.c a followed by 0 or more any char – anything, then .c

(Extended) Regular Expression Summary

Pattern	Maning	Example
c	Non-special, matches itself	'tom'
^	Start of line	'^ab'
$	End of line	'ab$'
.	Any single character	'.nodes'
[…]	Any single character in []	'[tT]he'
[^…]	Any single character not in []	'[^tT]he'
R*	Zero or more occurrences of R	'e*'
R?	Zero or one occurrences of R (egrep)	'e?'
R+	One or more occurrences of R (egrep)	'e+'
R1R2	R1 followed by R2	'[st][fe]'
R1|R2	R1 or R2 (egrep)	'the|The'

anchored

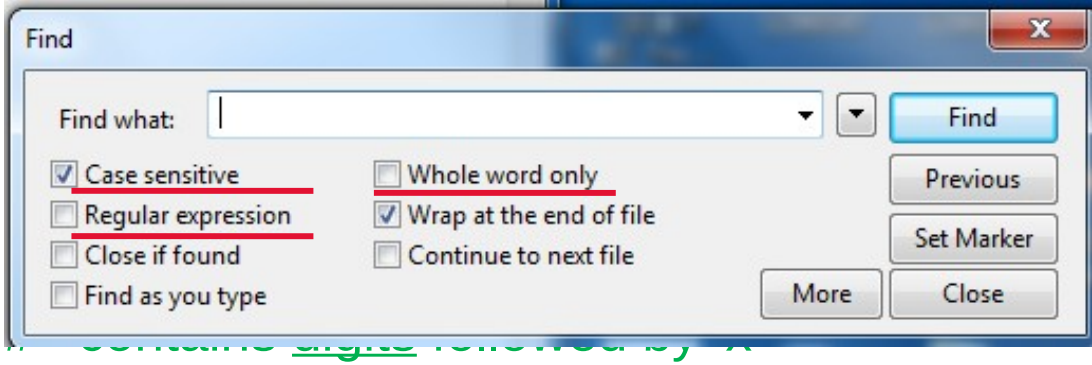
repetition

Utility	Kind of pattern that may be searched for
fgrep	fixed string only
grep	regular expression
egrep	extended regular expression

- Regular expression and extended expression maybe confusing.
- grep may behave differently in different shells.

- So for this course
  - Use grep -E or egrep take extended Regular Expression
  - Work on Bourne shell (sh) or Bourne again shell (bash)

Examples of Regex, grep



$ grep [0-9]x inputFile.txt

$ grep ^[tT]he inputFile.txt # begins with *the* or *The*

$ grep ^[a-z] inputFile.txt # begins with a lower case letter

$ grep .nd inputFile.txt # contains one any character followed by nd

$ grep [ab]nd$ inputFile.txt # ends with ‘and’ or ‘bnd’

$ grep -w W[ao]ng classlist # who have family name Wang or Wong?

YORK UNIVERSITÉ UNIVERSITY

The early question revisit

How to use grep to search lines that contain numbers?

```
$ grep   [0-9]   inputFile.txt          # or  grep [[:digit:]]  inputFile.txt
```

How to use grep to search lines that contain lower case letters?

```
$ grep   [a-z]    inputFile.txt          # or  grep [[:lower:]] inputFile.txt
```

Exit code of grep/egrep

Matching found: 0 No matching: 1 No such file: 2

```
$ grep Wang classlist
$ echo $?          # display its exit value.
0                  # indicates success.
$ grep Leung classlist
$ echo $?
1                  # indicates failure (not matching).
$ grep Wang  classlistXXX
grep: classlistXXX: No such file or directory
$ echo $?
2                  # indicates failure (not such a file).
```

Look for man
man grep | grep -w “exit”

Used in scripting

YORK UNIVERSITÉ UNIVERSITY