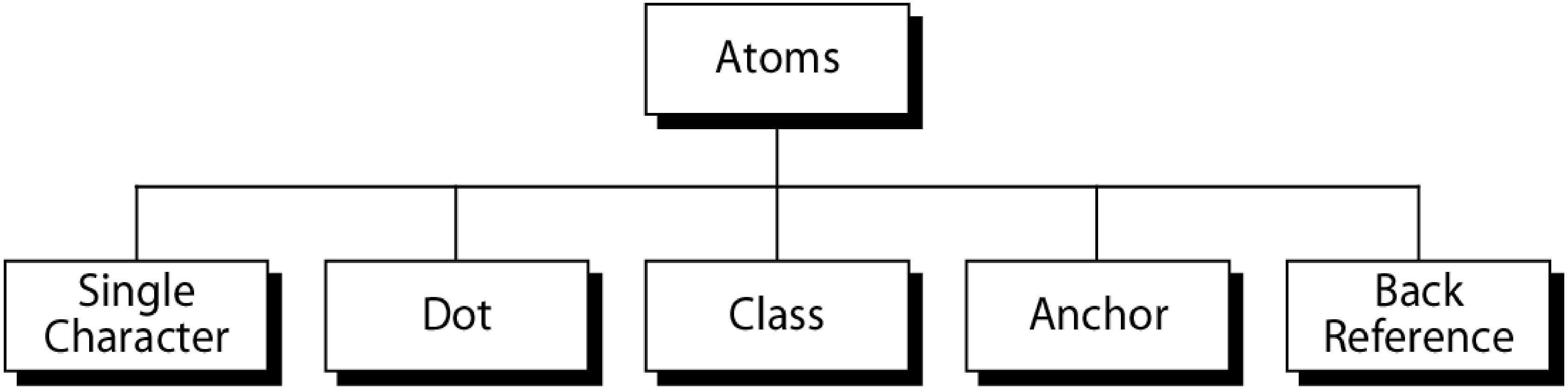
FILTERS USING REGULAR EXPRESSIONS – grep and sed

Regular expression

A regular expression (sometimes abbreviated to "regex") is a way for a computer user or programmer to express how a computer program should look for a specified pattern in text and then what the program is to do when each pattern match is found.

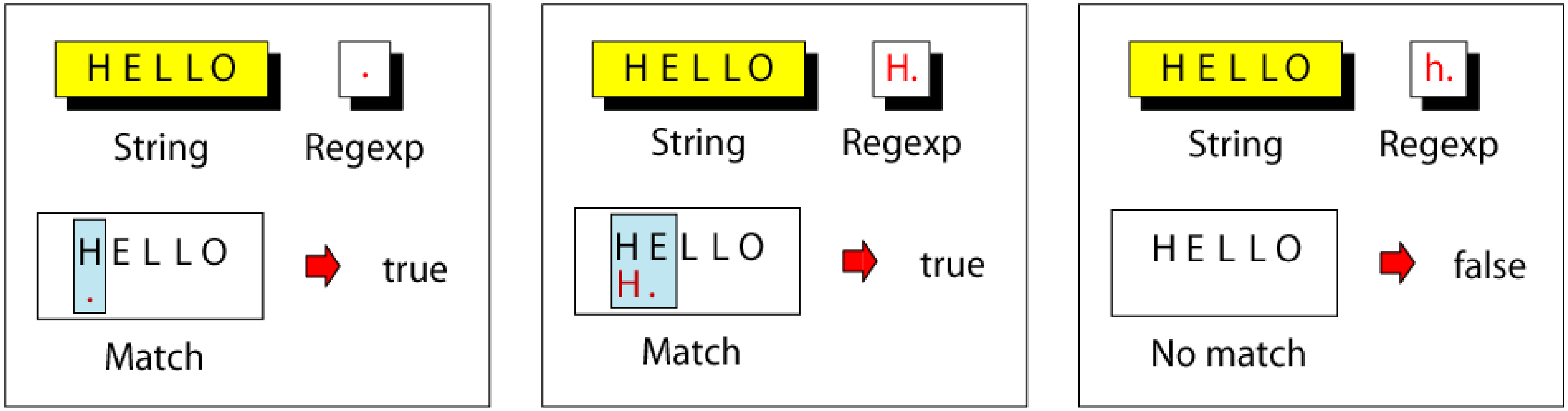
Atoms

An atom specifies what text is to be matched and where it is to be found.



Dot Atom

matches any single character except for a new line character (\n)

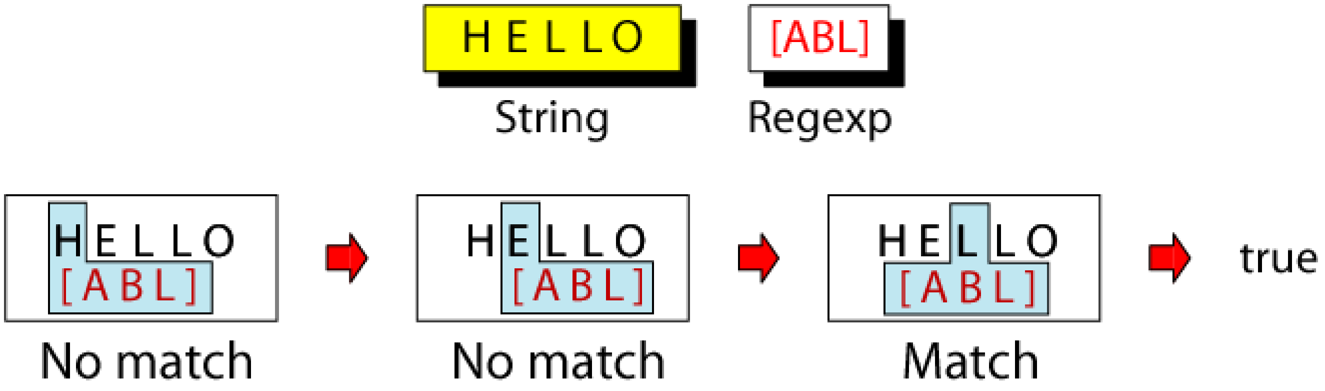


(a) Single-Character (b) Combination–True (c) Combination–False

Class Atom

matches only single character that can be any of the characters defined in a set:

<u>Example:</u> [ABC] matches either A, B, or C.



Notes:

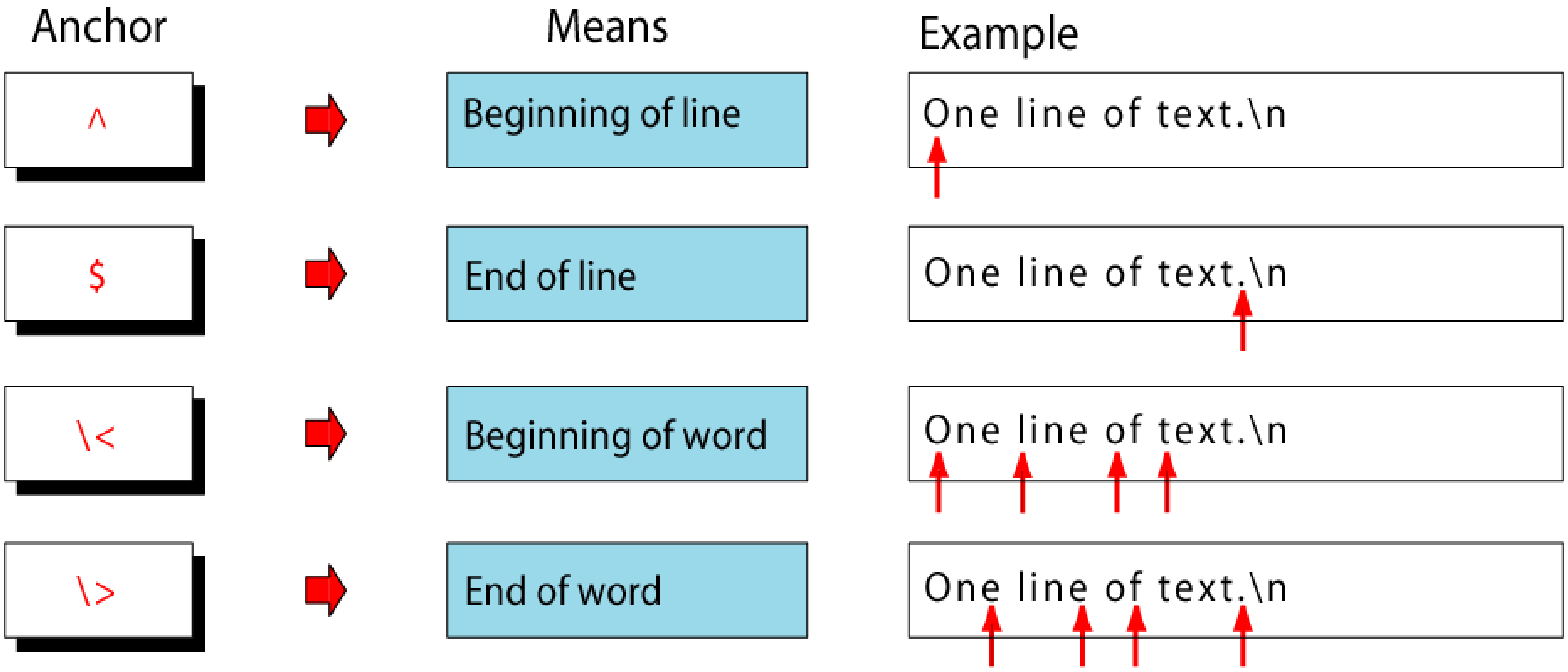
1) A range of characters is indicated by a dash, e.g. [A-Q]
2) Can specify characters to be excluded from the set, e.g.
   [^0-9] matches any character other than a number.

Example: Classes

RegExpr	Means	RegExpr	Means
[A-H]	[ABCDEFGH]	[^AB]	Any character except A or B
[A-Z]	Any uppercase alphabetic	[A-Za-z]	Any alphabetic
[0-9]	Any digit	[^0-9]	Any character except a digit
[[a]	[or a	[]a]	] or a
[0-9\-]	digit or hyphen	[^\^]	Anything except^

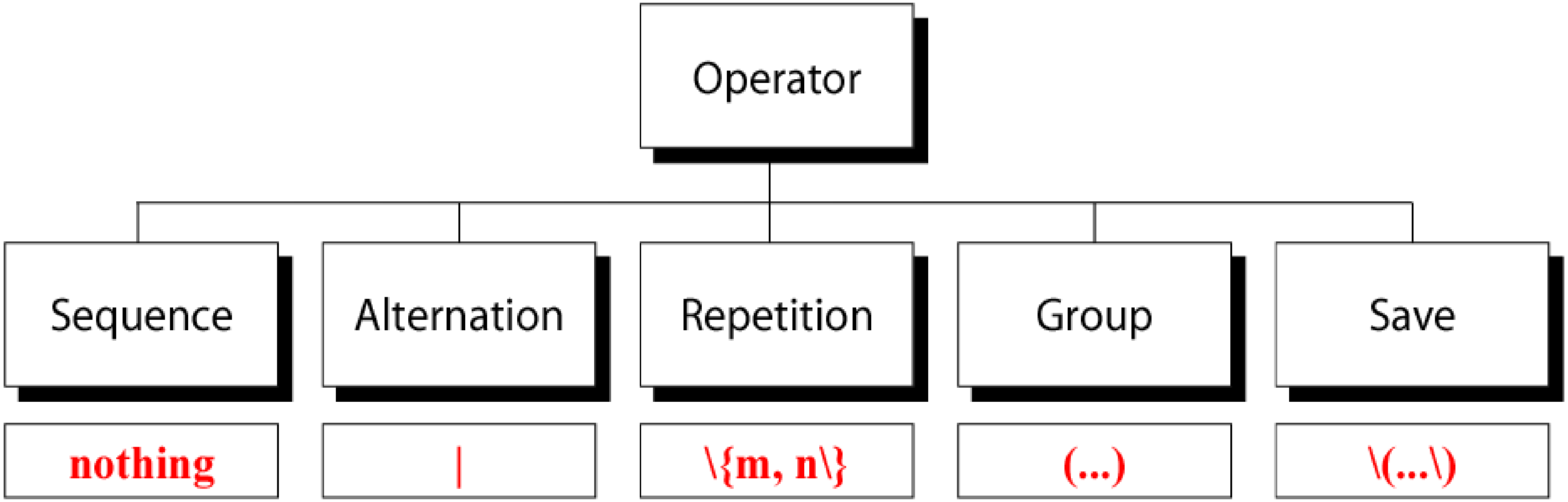
Anchors

Anchors tell where the next character in the pattern must be located in the text data.



Anchor	Means	Example
^	Beginning of line	One line of text.\n
$	End of line	One line of text.\n
\<	Beginning of word	One line of text.\n
\>	End of word	One line of text.\n

Operators



Operator

Sequence

nothing

Alternation

|

Repetition

\{m, n\}

Group

(...)

Save

\(...\)

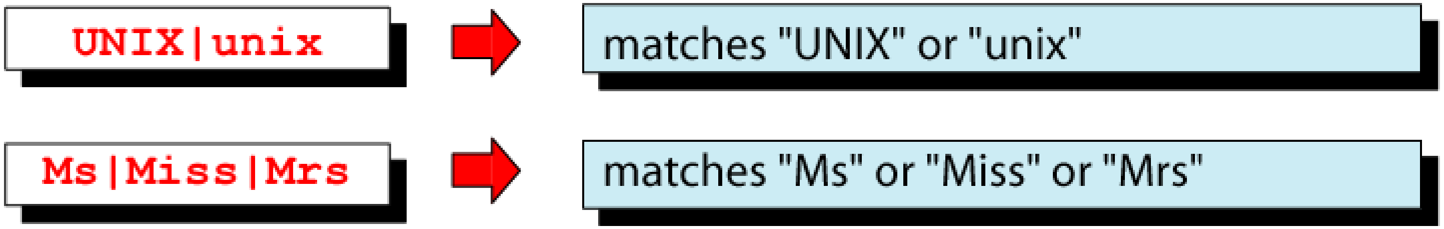
Sequence Operator

In a sequence operator, if a series of atoms are shown in a regular expression, there is no operator between them.

`dog`	→	matches the pattern "dog"
`a..b`	→	matches "a" , any two characters, and "b"
`[2-4][0-9]`	→	matches a number between 20 and 49
`[0-9][0-9]`	→	matches any two digits
`^$`	→	matches a blank line
`^.$`	→	matches a one-character line
`[0-9]-[0-9]`	→	matches two digits separated by a "-"

Alternation Operator: | or \|

operator (| or \|) is used to define one
or more alternatives



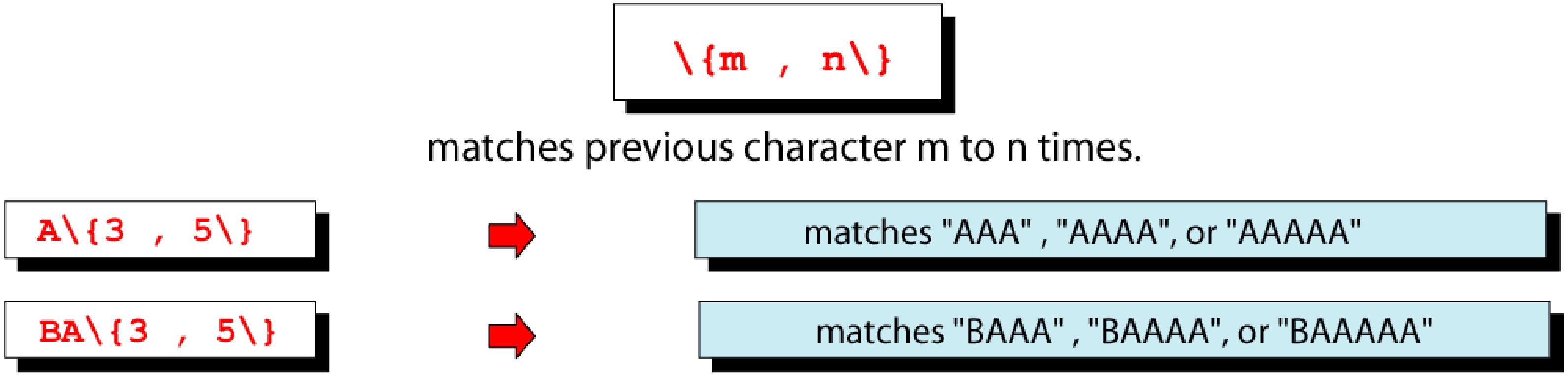
UNIX|unix → matches "UNIX" or "unix"

Ms|Miss|Mrs → matches "Ms" or "Miss" or "Mrs"

Note: depends on version of “grep”

Repetition Operator: \{...\}

The repetition operator specifies that the atom or expression immediately before the repetition may be repeated.



`\{m , n\}`

matches previous character m to n times.

`A\{3 , 5\}` → matches "AAA" , "AAAA", or "AAAAA"

`BA\{3 , 5\}` → matches "BAAA" , "BAAAA", or "BAAAAA"

Basic Repetition Forms

Formats

\{m\}	→	matches previous atom exactly m times
\{m, \}	→	matches previous atom m times or more
\{, n\}	→	matches previous atom n times or less

Examples

CA\{5\}	→	CAAAAA
CA\{3,\}	→	CAAA, CAAAA, CAAAAA, …
CA\{,2\}	→	C, CA, CAA

Short Form Repetition

Formats

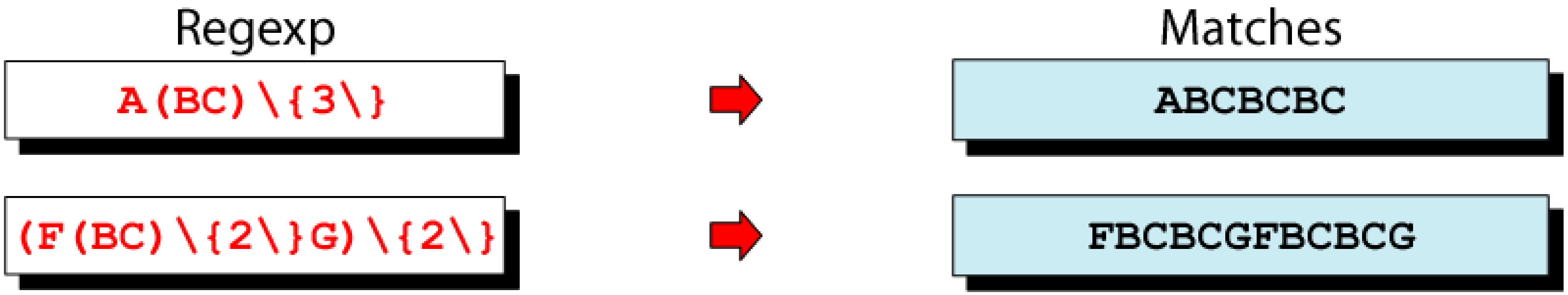
*	→	special case: matches previous atom zero or more times
+	→	special case: matches previous atom one or more times
?	→	special case: matches previous atom 0 or one time only

Examples

BA*	→	B, BA, BAA, BAAA, BAAAA, . . .
B.*	→	B, BA . . . BZ, BAA . . . BZZ, BAAA . . . BZZZ, . . .
.*	→	zero or more characters
.+	→	one or more characters
[0-9]?	→	zero or one digit

Group Operator

In the group operator, when a group of characters is enclosed in parentheses, the next operator applies to the whole group, not only the previous characters.



Regexp		Matches
`A(BC)\{3\}`	➡	ABCBCBC
`(F(BC)\{2\}G)\{2\}`	➡	FBCBCGFBCBCG

Note: depends on version of “grep”
use \(and \) instead

grep

- It scans the file / input for a pattern and displays lines containing the pattern, the line numbers or filenames where the pattern occurs
- It’s a command from a special family in UNIX for handling search requirements

  grep *options pattern filename(s)*

GREP DETAIL AND EXAMPLES

- grep is family of commands
  - grep

    common version
  - egrep

    understands extended REs

    (| + ? ( ) don’t need backslash)
  - fgrep

    understands only fixed strings, i.e. is faster
  - rgrep

    will traverse sub-directories recursively

COMMONLY USED “GREP” OPTIONS:

-c Print only a count of matched lines.

-i Ignore uppercase and lowercase distinctions.

-l List all files that contain the specified pattern.

-n Print matched lines and line numbers.

-s Work silently; display nothing except error messages. Useful for checking the exit status.

-v Print lines that do not match the pattern.

-e exp	specifies expression with this option
-x	matches pattern with entire line
-f file	takes pattrens from file, one per line
-E	treats pattren as an extended RE
-F	matches multiple fixed strings

```
grep “sales” emp.lst
```

- Patterns with and without quotes is possible
- Its generally safe to quote the pattern
- Quote is mandatory when pattren involves more than one word
- It returns the prompt in case the pattren can’t be located

```
grep president emp.lst
```

- When grep is used with multiple filenames, it displays the filenames along with the output

```
grep “director” emp1.lst emp2.lst
```

Where it shows filename followed by the contents

1. grep -i ‘agarwal’ emp.lst
2. grep -v ‘director’ emp.lst > otherlist

   wc -l otherlist will display 11 otherlist
3. grep –n ‘marketing’ emp.lst
4. grep –c ‘director’ emp.lst
5. grep –c ‘director’ emp*.lst

   will print filenames prefixed to the line count

6. grep –l ‘manager’ *.lst

   will display filenames *only*

7. grep –e ‘Agarwal’ –e ‘aggarwal’ –e ‘agrawal’ emp.lst

   will print matching multiple patterns

8. grep –f pattern.lst emp.lst

   all the above three patterns are stored in a separate file *pattern.lst*

BASIC REGULAR EXPRESSIONS

- It is tedious to specify each pattern separately with the -e option
- grep uses an expression of a different type to match a group of similar patterns
- if an expression uses meta characters, it is termed a regular expression
- Some of the characters used by regular expression are also meaningful to the shell

BASIC AND EXTENDED REGULAR EXPRESSIONS

(BRE & ERE)

BASIC REGULAR EXPRESSIONS

- It is tedious to specify each pattern separately with the -e option
- grep uses an expression of a different type to match a group of similar patterns
- if an expression uses meta characters, it is termed a regular expression
- Some of the characters used by regular expression are also meaningful to the shell

EXTENDED REGULAR EXPRESSIONS

Extended regular expressions (ERE)

- The + and ?
- Matching multiple patterns

BRE character subset

*****	Zero or more occurrences
g*	nothing or g, gg, ggg, etc.
.	A single character
.*	nothing or any number of characters
[pqr]	a single character p, q or r
[c1-c2]	a single character within the ASCII range represented by c1 and c2

The character class

- grep supports basic regular expressions (BRE) by default and extended regular expressions (ERE) with the –E option
- A regular expression allows a group of characters enclosed within a pair of [ ], in which the match is performed for a single character in the group

grep “[aA]g[ar][ar]wal” emp.lst

- A single pattern has matched two similar strings
- The pattern [a-zA-Z0-9] matches a single alphanumeric character. When we use range, make sure that the character on the left of the hyphen has a lower ASCII value than the one on the right

Negating a class (^) (caret)

THE *

* Zero or more occurrences of the previous character

g* nothing or g, gg, ggg, etc.

```
grep “[aA]gg*[ar][ar]wal” emp.lst
```

Notice that we don’t require to use –e option three times to get the same output!!!!!

THE DOT

A dot matches a single character

.* signifies any number of characters or none

```
grep “j.*saxena” emp.lst
```

^ and $

Most of the regular expression characters are used for matching patterns, but there are two that can match a pattern at the beginning or end of a line

^ for matching at the beginning of a line

$ for matching at the end of a line

```
grep “^2” emp.lst
```

Selects lines where emp_id starting with 2

```
grep “7…$” emp.lst
```

Selects lines where emp_salary ranges between 7000 to 7999

```
grep “^[^2]” emp.lst
```

Selects lines where emp_id doesn’t start with 2

When metacharacters lose their meaning

- It is possible that some of these special characters actually exist as part of the text
- Sometimes, we need to escape these characters

Eg: when looking for a pattern g*, we have to use \

To look for [, we use \[

To look for .*, we use \.*

EXTENDED RE (ERE)

- If current version of grep doesn’t support ERE, then use egrep but without the –E option
- -E option treats pattern as an ERE

+ matches one or more occurrences of the previous character

? Matches zero or one occurrence of the previous character b+ matches b, bb, bbb, etc.

b? matches either a single instance of b or nothing

These characters restrict the scope of match as compared to the *

```
grep –E “[aA]gg?arwal” emp.lst
```

```
# ?include +<stdio.h>
```

b+ matches b, bb, bbb, etc.

b? matches either a single instance of b or nothing

These characters restrict the scope of match as compared to the *

```
grep –E “[aA]gg?arwal” emp.lst
```

```
# ?include +<stdio.h>
```

The ERE set

ch+	matches one or more occurrences of character ch
ch?	Matches zero or one occurrence of character ch
exp1|exp2	matches exp1 or exp2
(x1|x2)x3	matches x1x3 or x2x3

SUMMARY

- BRE [ ], *, ., ^, $, \
- ERE ?, +, |, (,)
- sed: the stream editor

- THANK YOU