Regular Expression

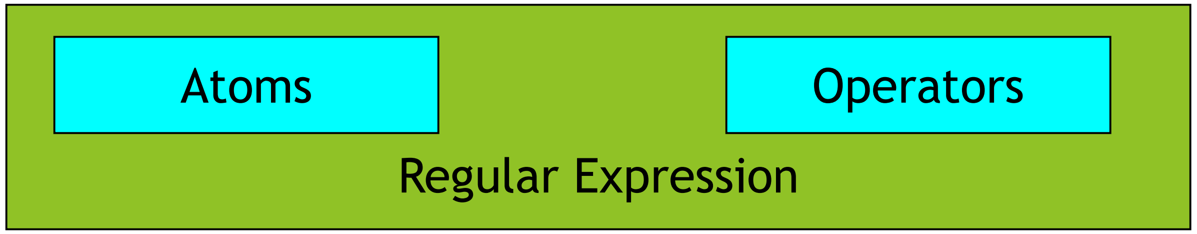
Amit Patel

Introduction

- Definition: Regular expression is a pattern ,consisting of a sequence of characters, that is matched against text.
- Unix evaluates text against the pattern to determine if the text and the pattern match. If they match, the expression is true and a command is executed. If they don't, the expression is false and command is not executed.
- Some of the most powerful UNIX utilities , "grep" and "sed", use regular expression.
- It is same like mathematical expression which contain operator and operands.

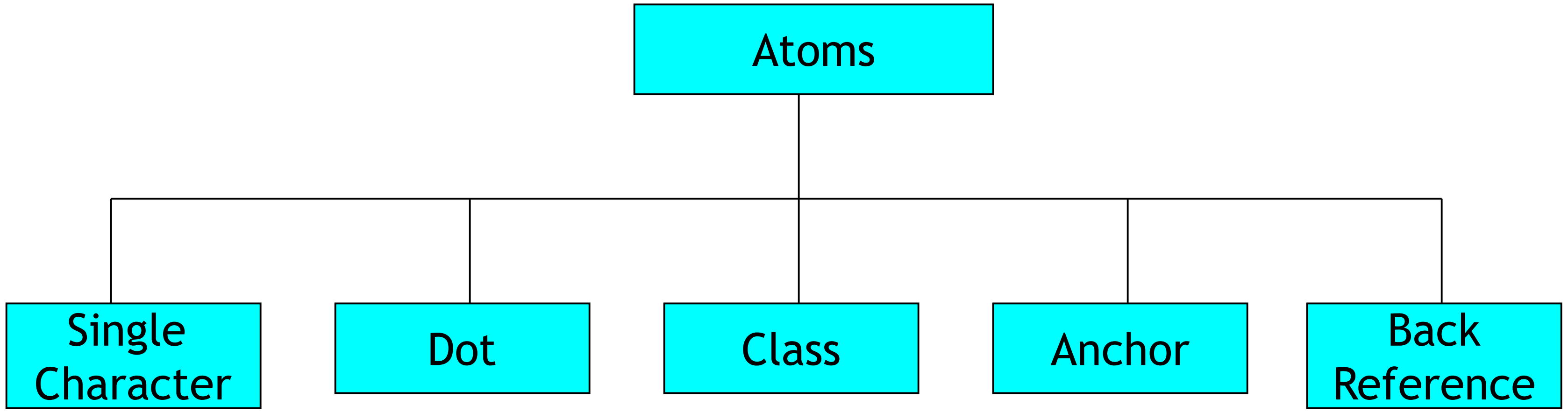
Introduction

- Regular expression is made of atoms and operators.
  - **Atoms** specifies what we are looking for and where in the text the match is to be made.
  - **Operator**, which is not required in all expressions, combines atoms into complex expressions.



Atoms

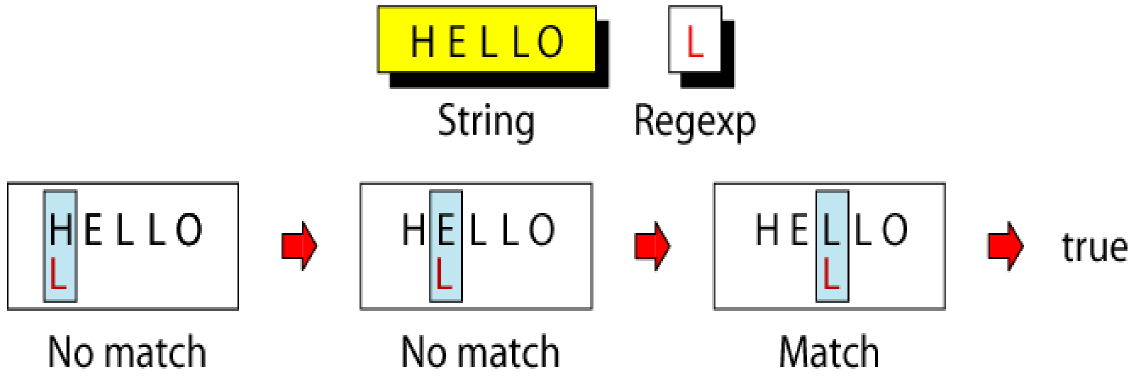
- An atom specifies what text is to be matched and where it is to be found.
- Atoms in regular expression can be of five types:



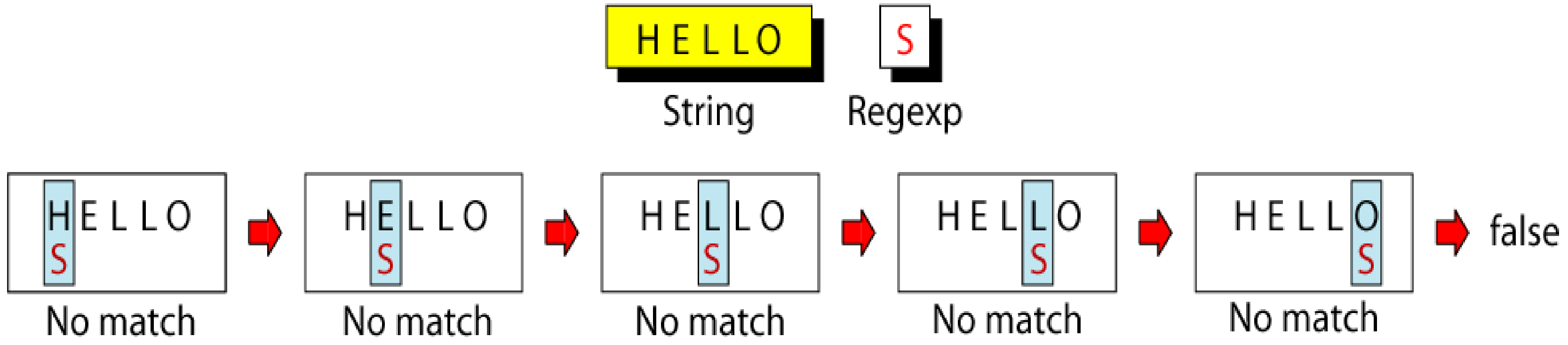
Atoms

Single Character

- The simplest atom is a single character.
- It matches itself.
- If a regular expression is made of one single character, that character must be somewhere in the text to make the pattern match successful.
- Matches single character anywhere in the text.
- If successful then return true. If fail then return false.



(a) Successful Pattern Match

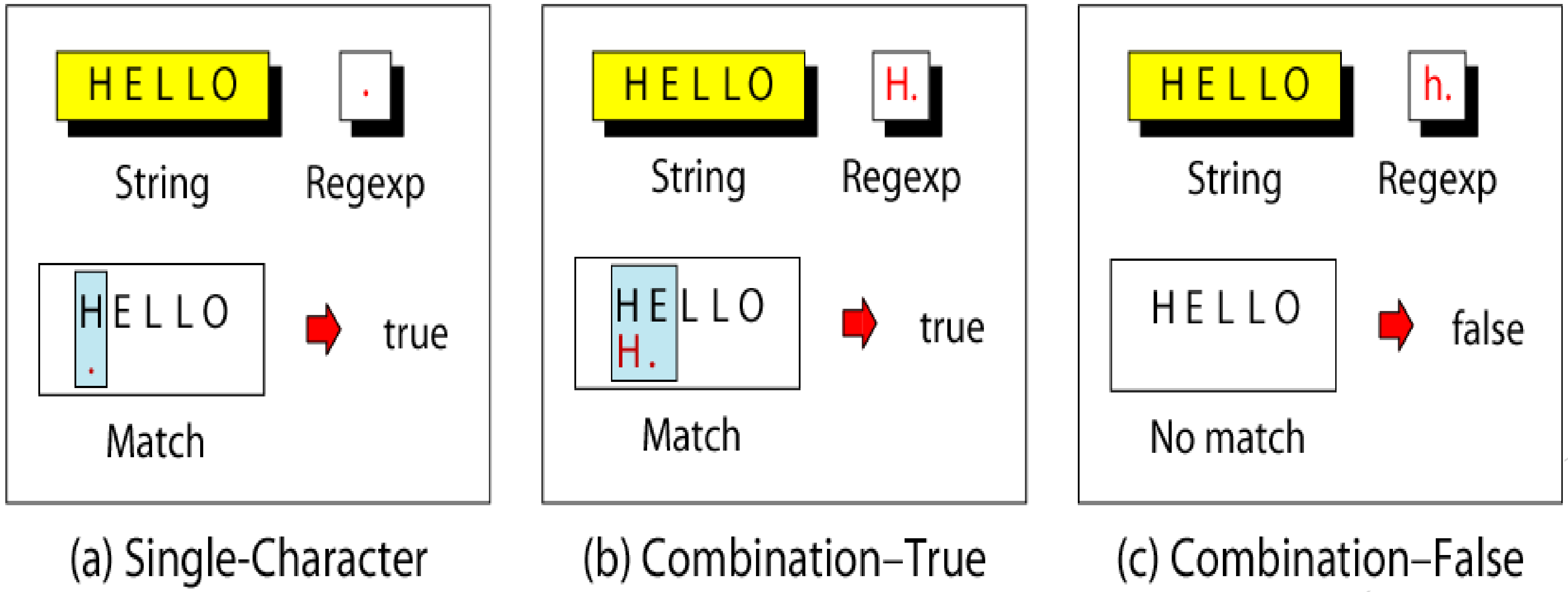


(b) Unsuccessful Pattern Match

Atoms

Dot

- A dot matches any single character except the new line character.
- It matches everything.
- It is more powerful when used with other character.



Atoms

Dot

- Its power in regular expression comes from its ability to work with other atoms to create expression.
- For example: H.
- This expression combine the single character atom, H, with the dot atom.
- It match any pair of characters where the first character is ‘H’.
- So it match : Ha,Hb,Hc….

Atoms

Class

- The class atoms defines a set of ASCII characters, any one of which may match any of the characters in the text.
- It matches only one single character from the set.
- The character set to be used in the matching process is enclosed in brackets.
- For Example: [ABC] matches either A, B, or C.
- It is very powerful expression component. Its power is extended with three additional tokens:
  - Range
  - Exclusion
  - Escape character

TOKEN	DESCRIPTION	EXAMPLE
Range	It is indicated by “ – “.	[a-d] indicates that the character a to d are all included in set.
Exclusion	It is indicated by “ ^ “.	[^aeiou] specifies any character other than “a,e,I,o,u”
Escape Character	It is indicated by “ \ ”. It is used when the matching characters is one of the other two tokens.	[aeiou\-] specifies to match vowel or a dash. The Escape character “\” indicates that the dash is a character not a range token

RegExpr	Means
[A-H]	[ABCDEFGH]
[A-Z]	Any uppercase alphabetic
[0-9]	Any digit
[[a]	[or a
[0-9\-]	digit or hyphen

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

Atoms

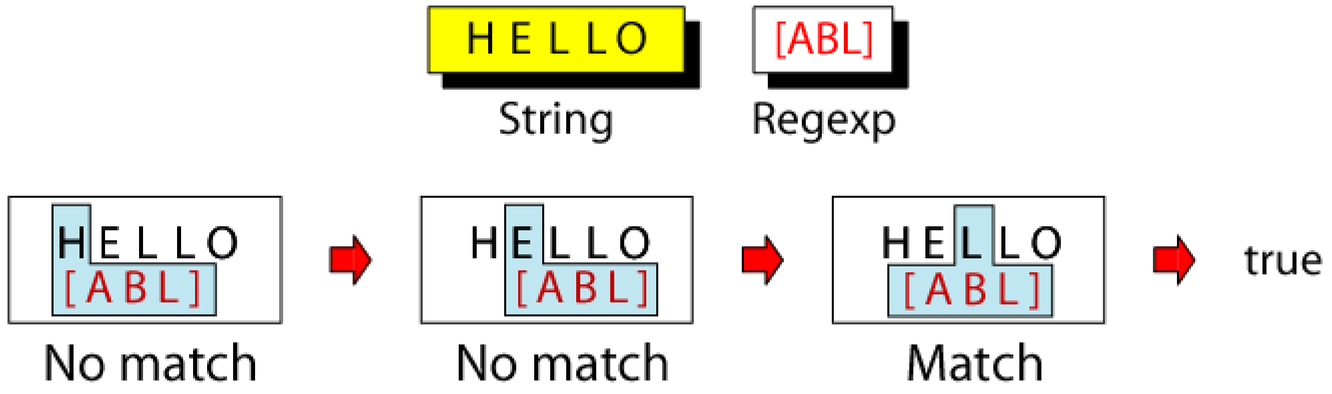
- List out line from file which contain “Vivek” or “vivek”.
- List line which contain any number.
- List line which contain not contain any digit.
- Search word which start with “T” and end with “K” with max. 5 character.
- Search for a word which is having three letters in it and starts with x and ends with m.
- Search words do not contain ‘ac’ in a file.
- Search for a ‘[ ] \ ? *‘ in a file

Atoms

Class

- A range of characters is indicated by a dash, e.g. [A-Q]
- Can specify characters to be excluded from the set.

  e.g. [^0-9] matches any character other than a number.

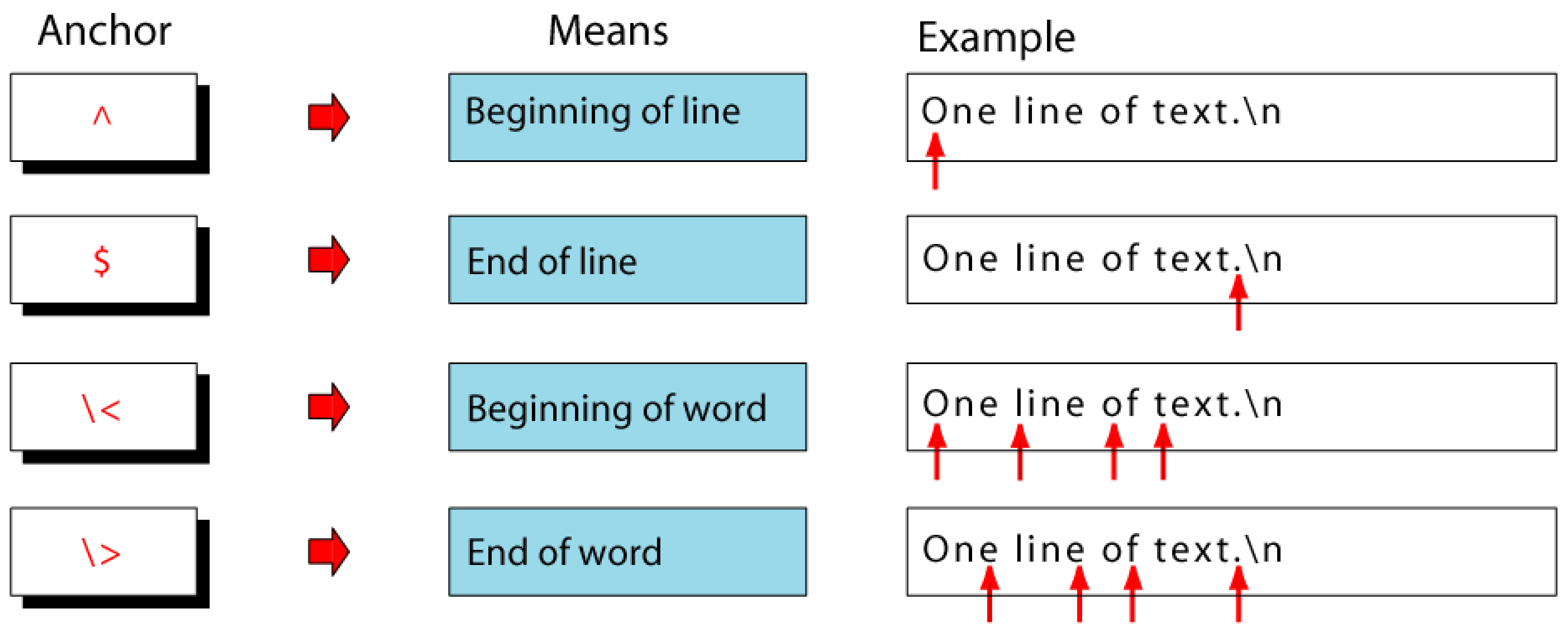


Atoms

Anchors

- Anchors are atoms that are used to line up the pattern with a particular part of a string
- Anchors are not matched to the text, but define where the next character in the pattern must be located in the text.
- There are four type of anchors.

Atoms



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

Atoms

Example: List out file which start with character ‘I’

```
Cat  test.txt
```

```
My name is Amit Patel.
I am working as a Assistant Professor in BCA department in VTCBB.
If i talk about my education...
I have completed my mca from SRIMCA college in 2007 with 55.30%.
Now time to talk about past work experiences...
I have 3 years of experience as Network Engineer and 6 Month as SW Tester.
```

```
Grep ^I  test.txt
```

```
I am working as a Assistant Professor in BCA department in VTCBB.
If i talk about my education...
I have completed my mca from SRIMCA
I have 3 years of experience as Network Engineer and 6 Month as SW Tester.
I joied this institute in June 2012.
```

Atoms

Example

- List line which end with “ you.”
- Display all lines of file using “grep”
- List line which contain only one character.
- List line which have no any character.
- List line which have only “I” character.
- List line which have exactly 4 character which start with “H”.
- List all line which end with “$”.
- How many word of 5 letters in file?
- How many different words of exactly 5 letters are there in exatext1?
- Count all word ending with “ing”

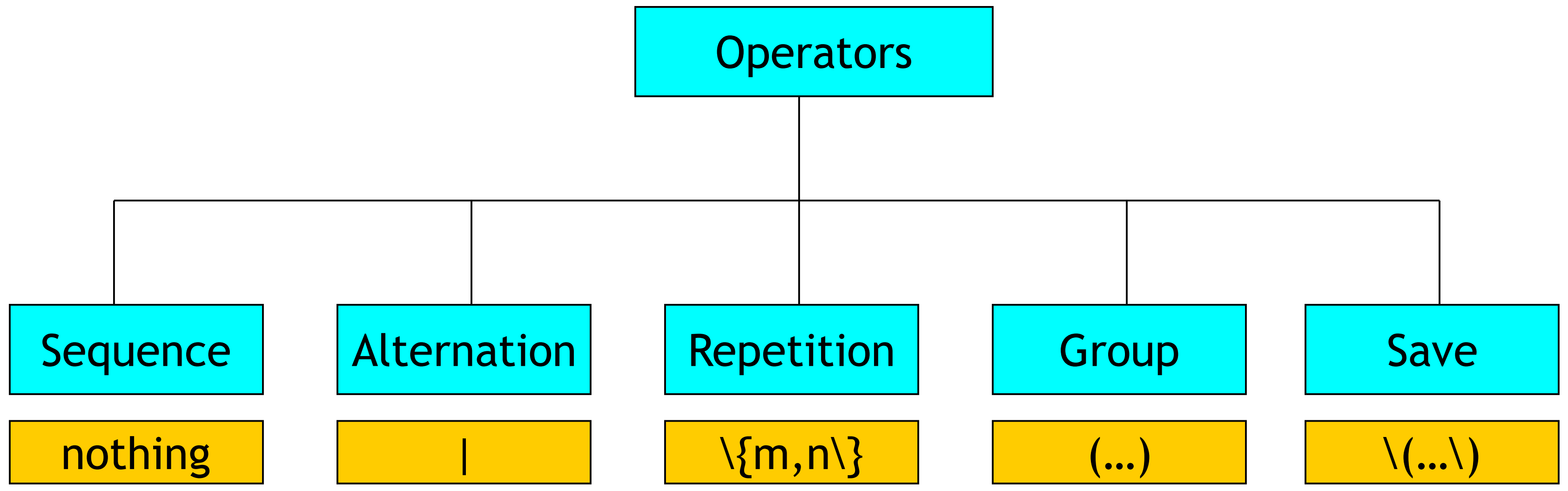
Atoms

Back References

- We can temporarily save text in one of nine save buffers.
- This can be done using a back reference.
- A back reference is coded using the escape character and a digit in the range of 1 to 9.
- Example: \1 \2 ….. \9
- It is used to match text in the current or designated buffer with text that has been saved in one of the system's nine buffers.

Operator

- To make regular expression more powerful we can combine atoms with operators.
- Regular expression operators combine regular expression atoms.
- Regular expression operator can be combine into 5 category.



Operator

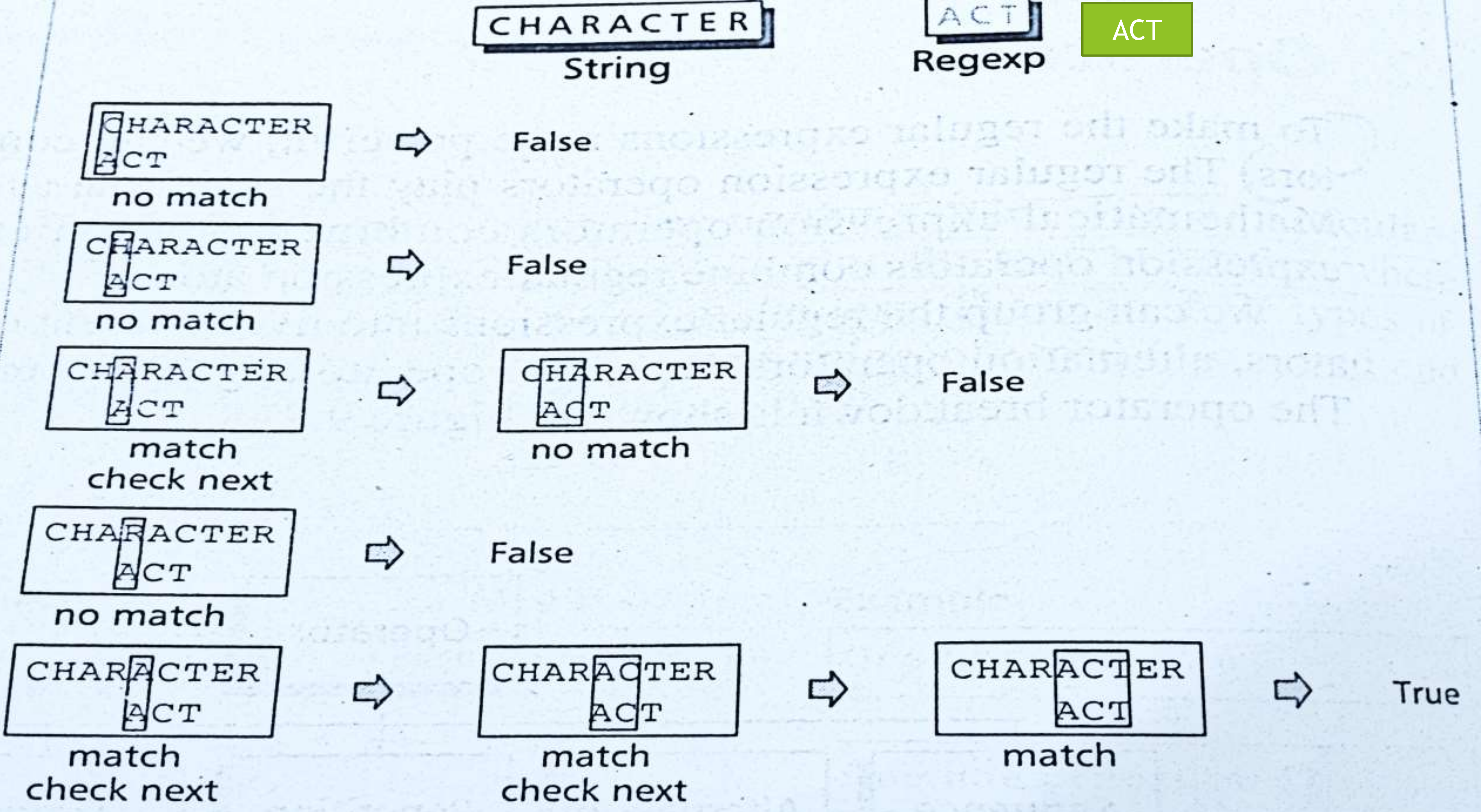
Sequence

- The sequence operator is nothing.
- Means series of atoms such as series of characters are shown in regular expression.
- It implies that there is an invisible sequence operator between them.

Operator

`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 "-"

ACT


CHARACTER
String
ACT
Regexp
CHARACTER
ACT
no match
False
CHARACTER
ACT
no match
False
CHARACTER
ACT
match
check next
CHARACTER
ACT
no match
False
CHARACTER
ACT
no match
False
CHARACTER
ACT
match
check next
CHARACTER
ACT
match
check next
CHARACTER
ACT
match
True

Operator

Alternation (| or \|)

- An alternation operator is used to define one or more alternatives.
- operator (| or \|) is used to define one or more alternatives
- For example: If we want to select between A or B, we would code the regular expression as A | B.
- It is used with single atoms, but it is also use for selecting between two or more sequences of characters or groups of characters.

Operator

UNIX|unix → matches "UNIX" or "unix"

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

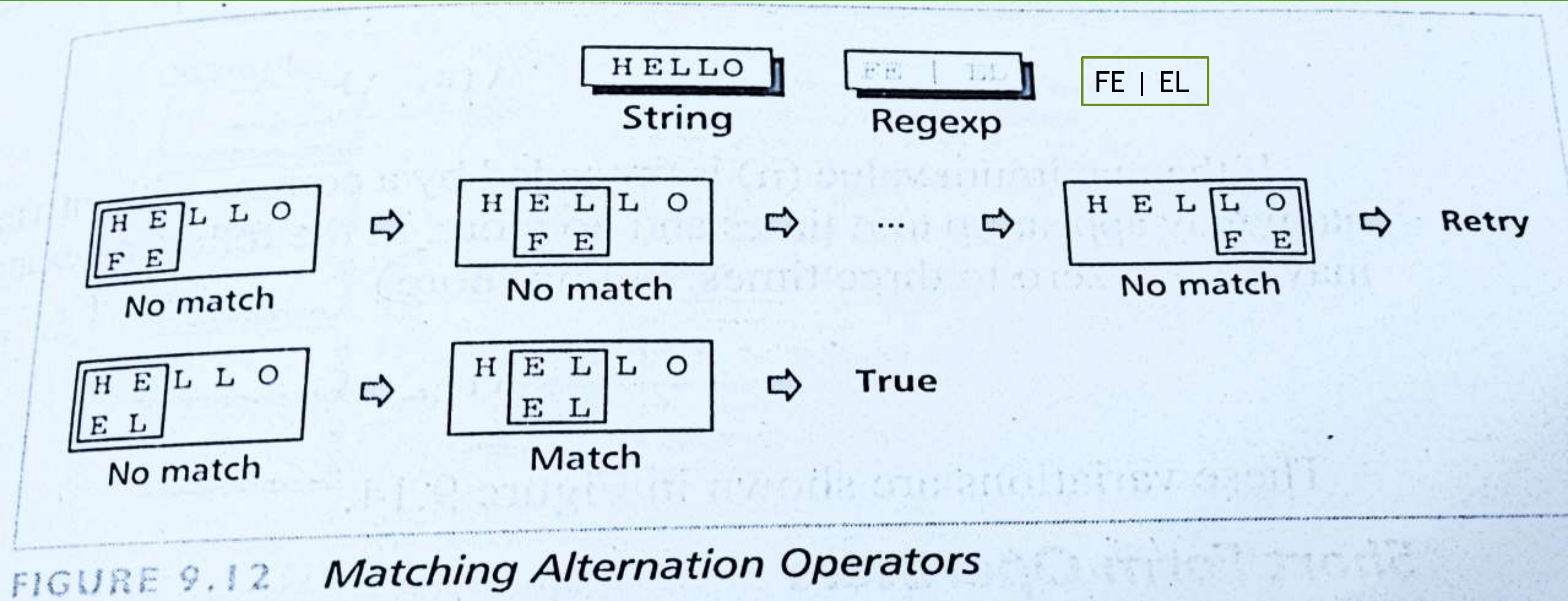


FIGURE 9.12 *Matching Alternation Operators*

Operator

Repetition

- The repetition operator is a set of escaped braces \{….\} that contains two numbers separated by comma.
- It specify that the atom or expression immediately before the repetition may be repeated.
- \{ m , n \} : matches previous character m to n times.
- The first number, m, indicates the minimum required times the previous atom must appear in the text
- The second number, n, indicates the maximum number of times it may appear.

Operator

`\{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"

Operator

Basic Repetition Form

- The “m” and “n” values are optional although at least one must be present.
- If any one repetition value (m) is enclosed, {m}, the previous atom must be repeated exactly “m” times.
- If the minimum value (m) is followed by a comma without a maximum value, {m,},the previous atom must be present at least “m” times, but it may appear more than “m” times.
- If the maximum value (n) is preceded by a comma without a minimum, {,n},the previous atom appear up to “n” times and no more.

Operator

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`

Operator

Short Form Operator

- Three form of repetition are common : *, +, ?

FORM	MEANING
*	Repeat an atom zero or more times. Same as \{0,\}
+	Repeat an atom one or more times. Same as \{ 1 , \}
?	Repeat the pattern zero or one time only. Same as \{ 0 , 1 \}

Operator

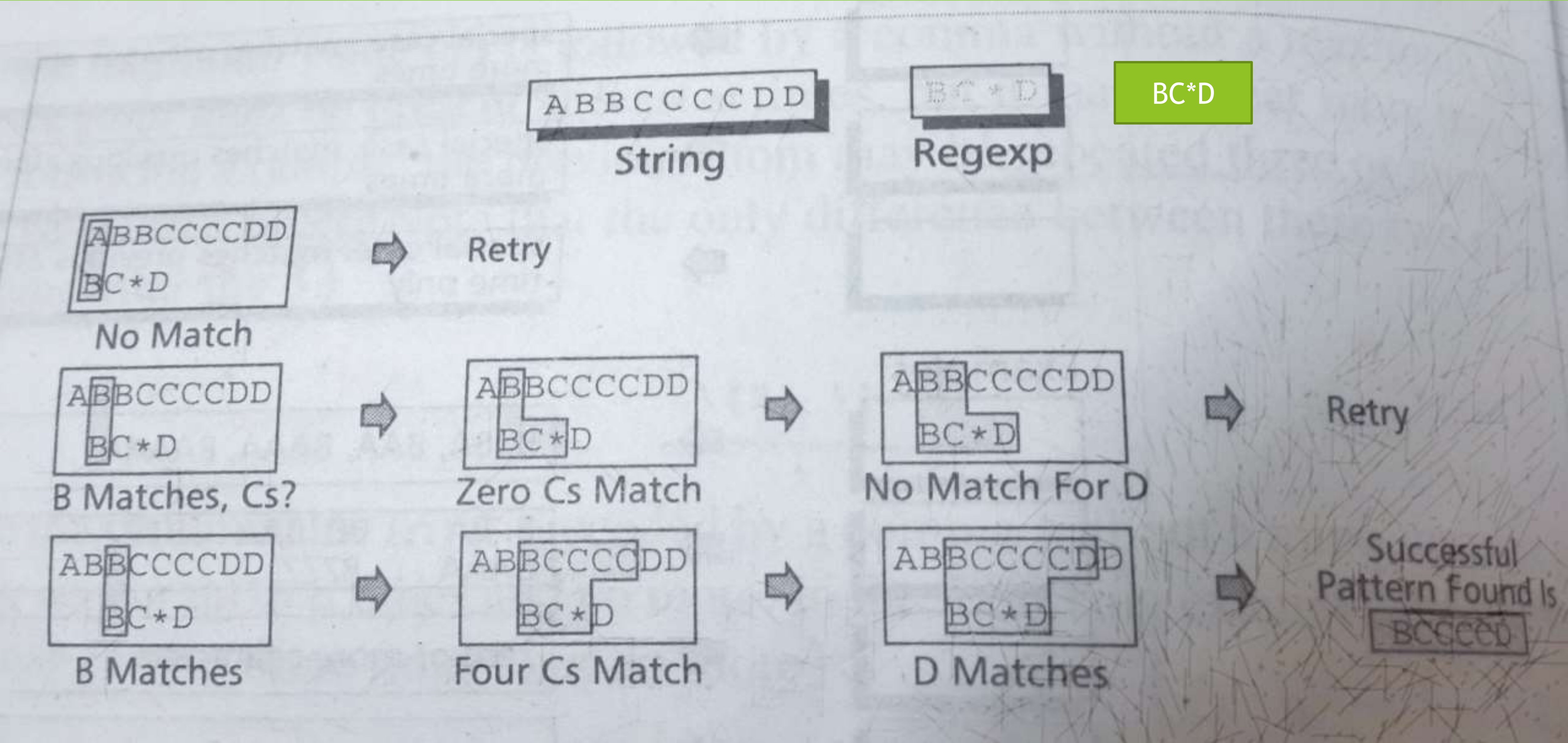
Formats

*	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

Operator

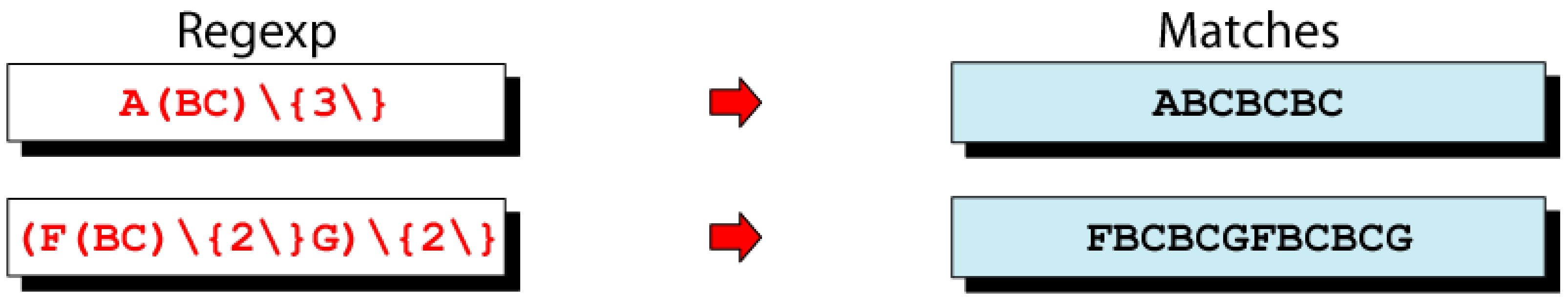


BC*D

Operator

Group Operator

- The group operator is a pair of opening and closing parentheses.
- When group of operator is enclosed in parentheses, the next operator applies to the whole group, not only to the previous character.
- A(BC)\{3\} : ABCBCBC



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

Operator

Save

- The save operator, which is set of escaped parentheses, \(…\), copies a matched text string to one of nine buffers for later reference.
- Within an expression, the first saved text is copied to buffer 1, after second saved text is coped to buffer 2…… once text has been saved, it can be referred by using a back reference. Using \1 \2 \3 ….

NOTE

- The "*" has a meaning in a regular expression that is different than its meaning as a wildcard.
- In a regular expression, it means repeat the previous character or group of characters zero or more times. As a wildcard in filename, it means zero or more characters.
- The regular expression usage means that it cannot be used by itself, it must be preceded by an atom or a group of atoms.

NOTE

- The “?” has a meaning in a regular expression that is different than its meaning as a wildcard.
- In a regular expression, it means repeat the previous character or group of characters zero or one times. As a wildcard in filename, it means one characters.
- The regular expression usage means that it cannot be used by itself, it must be preceded by an atom or a group of atoms.

grep

- Full Form: Global Regular Expression Print
- It is a family of program that is used to search the input file for all lines that match a specified regular expression and write them to the standard output file.
- There are following option available in grep.

Operator

grep option regexp file list

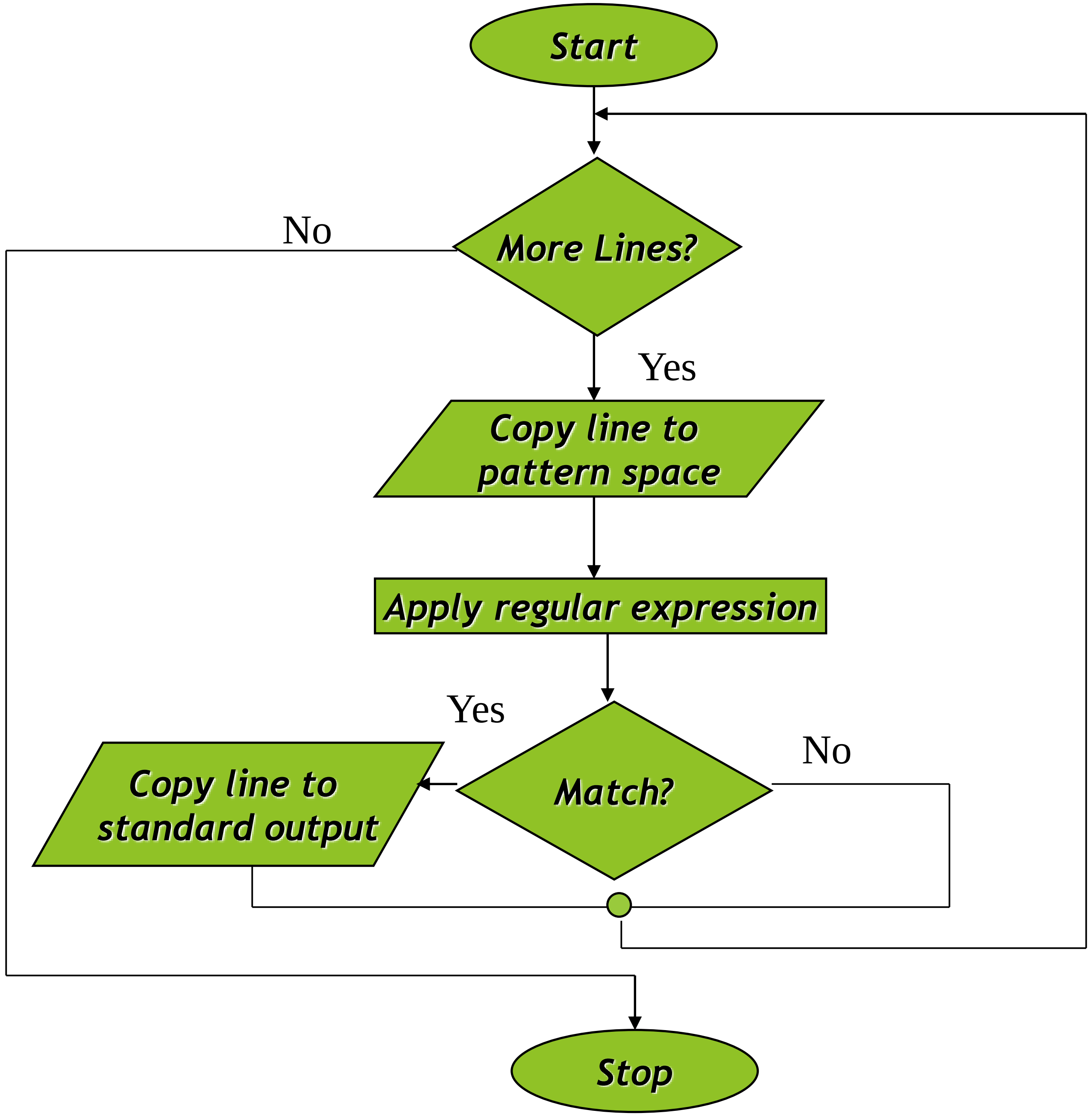
OPTION	MEANING
-b	Print Block numbers.
-c	Print only match count.
-I	Ignore upper lower case/
-l	Print file with at least one match.
-n	Print line numbers.
-s	Silent mode; no output.
-v	Print lines that don’t match.
-x	Print lines that match.
-f file	Expression are in file

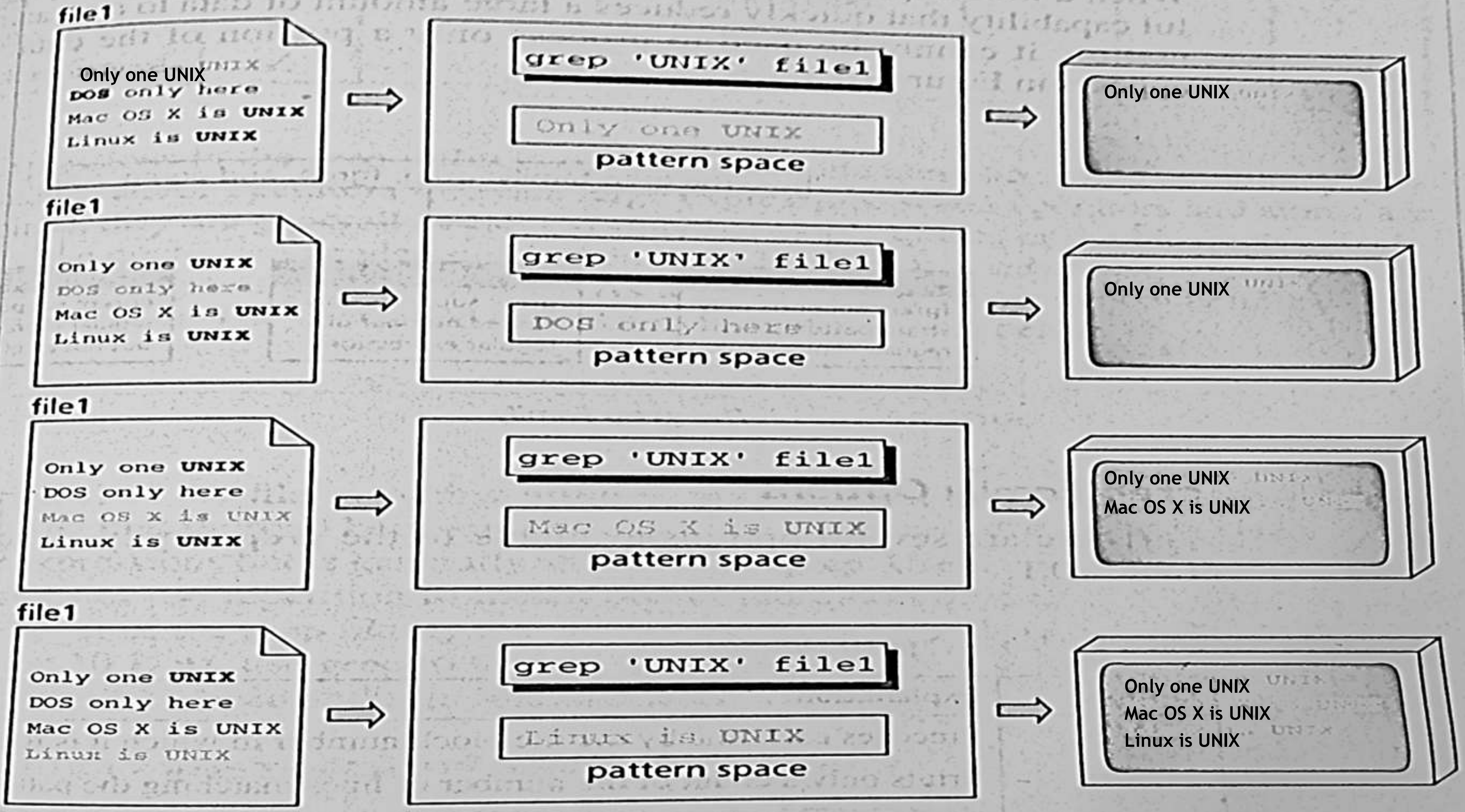
Operation

- grep is a search utility; it can search only for the existence of a line that matches a regular expression
- The only action that grep can perform on a line is to send it to standard output. If line do not match RE then, it do not print
- The line selection is based on RE. The line number or other criteria can not be used for selection
- grep is a filter. Can be used both sides of pipe.

Opearation

- For each line in the standard input or file, grep perform following operation.
  - Copies the next input line into the pattern space. The pattern space is a buffer that can hold only one text line.
  - Applies the regular expression to the pattern space.
  - If there is a match, the line is copied from the pattern space to the standard outuput.
- “grep” utility repeat these step on each line in the input.


Start
More Lines?
No
Yes
Copy line to pattern space
Apply regular expression
Yes
Match?
No
Copy line to standard output
Stop



```
file1:
Only one UNIX
DOS only here
Mac OS X is UNIX
Linux is UNIX
```

```
grep 'UNIX' file1
```

Pattern space: Only one UNIX

```
Only one UNIX
```

Pattern space: DOS only here

```
Only one UNIX
```

Pattern space: Mac OS X is UNIX

```
Only one UNIX
Mac OS X is UNIX
```

Pattern space: Linux is UNIX

```
Only one UNIX
Mac OS X is UNIX
Linux is UNIX
```

Limitation

- Cannot be used to add, delete or change a line.
- Cannot be used to print only part of a line.
- Cannot read only part of a file.
- Cannot select a line based on the content of the previous or the next line .

grep

- Create a two file

File	File1
$ cat file OM,20,SURAT,BCA,258307 SAI,19,BARDOLI,BBA,245678 RAM,5,NAVSARI,BBA,222434 VATSHAL,49,BARODA,BCA,258783 AADI,200,SURAT,BCA,264416	$ cat file1 NEEL,12,VYARA,BCA,248317 PREM,45,BARDOLI,BCA,234567 ARYAN,55,NAVSARI,BBA,332435 DIYA,68,surat,BBA,268683 MANASI,27,SURAT,BCA,274905

grep

- Example 1: List the record which is coming from "SURAT"

```
$ grep "SURAT" file
OM,20,SURAT,BCA,258307
AADI,200,SURAT,BCA,264416
```

- Example 2: List the record which is coming from "SURAT" from file and file1

```
$ grep "SURAT" file file1
fiel: OM,20,SURAT,BCA,258307
file: AADI,200,SURAT,BCA,264416
file1: MANASI,27,SURAT,BCA,274905
```

grep

Ignoring case (-i)

- Example 3: List the record which is coming from “surat” or “SURAT” from file1.

```
$ grep  -I “surat” file file2

DIYA,68,surat,BBA,268683

MANASI,27,SURAT,BCA,274905
```

grep

Deleting lines (-v)

- Example 4: Count student which is not coming from surat.

```
$ grep  -v "surat" file file2 | wc -l
7
```

Displaying Line numbers (-n) : Display line number which contain search pattern.

- Example 5: List the record which is coming from "surat" or "SURAT" from file1 with position in file.

```
$ grep  -in "surat"  file2
4:DIYA,68,surat,BBA,268683
5:MANASI,27,SURAT,BCA,274905
```

grep

Counting lines containing pattern (-c)

- Example 6: Count student which is coming from surat.

```
$ grep  -c “surat” file file2

1
```

If we pass multiple file:

```
$ grep  -ci “surat” file file*

file: 2

file2:2
```

grep

Displaying Filename (-l): -l option display only the name of files containing te pattern

- Example 7: Display file name which contain record for student coming from “surat”

```
$ grep  -il “surat”  *
file
file2
```

Multiple matching pattern (-e) :We can match the multiple pattern with this option.

- Example 4: List the record which is coming from “surat” or “SURAT” from file1 with line number.

```
$ grep  -e “surat”  -e “SURAT” file2
DIYA,68,surat,BBA,268683
MANASI,27,SURAT,BCA,274905
```

Example: grep with \< \>

$ cat grep-datafile

```
northwest    NW    Charles Main          300000.00
western      WE    Sharon Gray           53000.89
southwest    SW    Lewis Dalsass         290000.73
southern     SO    Suan Chin             54500.10
southeast    SE    Patricia Hemenway     400000.00
eastern      EA    TB Savage             440500.45
northeast    NE    AM Main Jr.           57800.10
north        NO    Ann Stephens          455000.50
central      CT    KRush                 575500.70
Extra   [A-Z]****[0-9]..$5.00
```

grep

- Example: Print all lines beginning with either a “w” or an “e”.

```
$ grep '^[we]' grep-datafile
western     WE     Sharon Gray        53000.89
eastern     EA     TB Savage         440500.45
```

- Example: Print the lines that contain either the expression “NW” or the expression “EA”

```
$ grep 'NW\|EA' grep-datafile
northwest      NW     Charles Main       300000.00
eastern        EA     TB Savage          440500.45
```

grep

- Example: Print all lines ending with a period and exactly two zero numbers.

```
$ grep '\.00$' grep-datafile
northwest     NW     Charles Main          300000.00
southeast     SE     Patricia Hemenway      400000.00
Extra [A-Z]****[0-9]..$5.00
```

- Example: Print all lines containing one or more 3's.

```
$ grep '3\+'  grep-datafile   OR $ egrep '3+' grep-datafile  OR $ grep '3\{1,\}' grep-datafile
northwest      NW     Charles Main          300000.00
western        WE     Sharon Gray           53000.89
southwest      SW     Lewis Dalsass          290000.73
```

grep

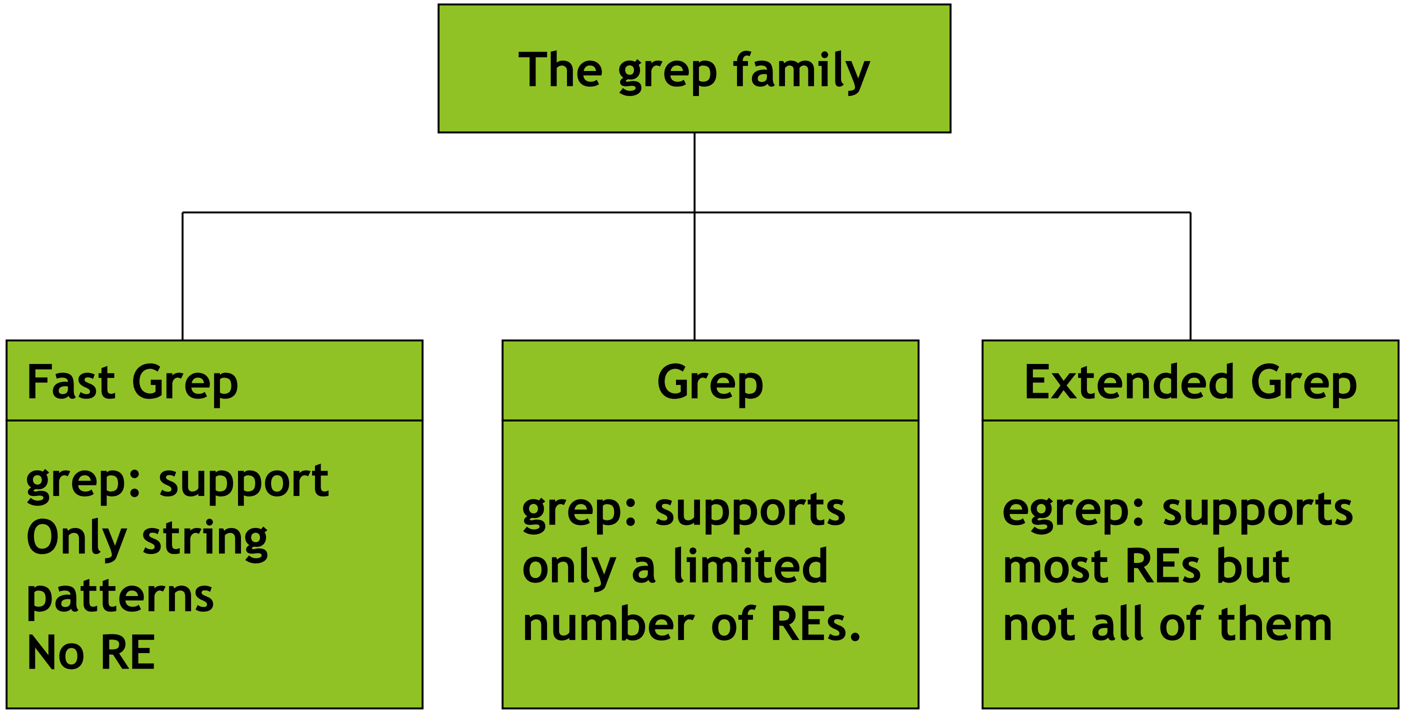
- Example: Print only all directory file in current directory.

```
$ ls -l | grep '^d'
drwxr-xr-x   2 krush    csci          512 Feb  8 22:12 assignments
drwxr-xr-x   2 krush    csci          512 Feb  5 07:43 feb3
drwxr-xr-x   2 krush    csci          512 Feb  5 14:48 feb5
drwxr-xr-x   2 krush    csci          512 Dec 18 14:29 grades
drwxr-xr-x   2 krush    csci          512 Jan 18 13:41 jan13
```

- Example: Print total number of directory in current directory.

```
$ ls -l | grep -c '^d'
10
```

grep Family


The grep family
Fast Grep
grep: support Only string patterns No RE
Grep
grep: supports only a limited number of REs.
Extended Grep
egrep: supports most REs but not all of them

grep Family Expressions

Atoms	Grep	Fgrep	Egrep
Character	✓	✓	✓
Dot	✓		✓
Class	✓		✓
Anchors	✓		^ $
Back Reference	✓		

grep Family Expressions

Operators	Grep	Fgrep	Egrep
Sequence	✓	✓	✓
Repetition	**All but ?**		* ? +
Alternation			✓
Group			✓
Save	✓		

egrep

- Full Form: Extended Global Regular Expressions Print.
- The 'E' in egrep means treat the pattern as a regular expression. "Extended Regular Expressions" abbreviated 'ERE' is enabled in egrep.
- egrep which is the same as “grep –E”
- It treats +, ?, |, (, and) as meta-characters.
- In basic regular expressions (with grep), the meta-characters ?, +, {, |, (, and) lose their special meaning.
- If you want grep to treat these characters as meta-characters, escape them \?, \+, \{, \|, \(, and \).
- Following is the meaning of all these meta characters in egrep.

NOTE: Use egrep, when command contain - ?, +, {, |, (,) - metacharacter.

egrep

- + : Matches one or more occurrence of the previous character.
- ? : Matches zero or one occurrence of the previous character.
- | : Matches multiple pattern
- ( ): Group pattern.

egrep

- For example: here grep uses basic regular expressions where the plus is treated literally, any line with a plus in it is returned.

```
$ grep "+" myfile.txt
```

- egrep on the other hand treats the "+" as a meta character and returns every line because plus is interpreted as "one or more times".

```
$ egrep "+" myfile.txt
```

- Here every line is returned because the + was treated by egrep as a meta character. normal grep would have searched only for lines with a literal +.

Example: egrep with +

```
$cat grep-datafile
northwest      NW     Charles Main         300000.00
western         WE     Sharon Gray          53000.89
southwest      SW     Lewis Dalsass        290000.73
southern       SO     Suan Chin            54500.10
southeast      SE     Patricia Hemenway    400000.00
eastern        EA     TB Savage            440500.45
northeast      NE     AM Main Jr.          57800.10
north          NO     Ann Stephens          455000.50
central        CT     KRush                 575500.70
Extra [A-Z]****[0-9]..$5.00
```

Print all lines containing one or more 3's.

Note: grep works with \+

```
$ egrep '3+' grep-datafile

northwest      NW     Charles Main         300000.00

western        WE     Sharon Gray          53000.89

southwest      SW     Lewis Dalsass        290000.73
```

Example: egrep with RE: ?

Print all lines containing a 2, followed by zero or one period, followed by a number.

```
$ egrep '2\.?[0-9]' grep-datafile
southwest       SW      Lewis Dalsass           290000.73
```

Note: grep works with \?

Example: egrep with ( )

Print all lines containing one or more consecutive occurrences of the pattern “no”.

```
$ egrep '(no)+' grep-datafile
northwest       NW      Charles Main            300000.00
northeast       NE      AM Main Jr.             57800.10
north           NO      Ann Stephens            455000.50
```

Note: grep works with \(\) \+

Example: egrep with (a|b)

Print all lines containing the uppercase letter “S”, followed by either “h” or “u”.

```
$ egrep 'S(h|u)' grep-datafile
western      WE     Sharon Gray        53000.89
southern     SO     Suan Chin          54500.10
```

Note: grep works with \(\) \|

egrep

- For example: Write a command to print line which is start with capital leter and end with exclamation mark.

```
$ egrep '^[A-z].*!$' myfile.txt
```

- This is relatively complex expression. Here we have 3 expression in one pattern.
- The first expression start at the beginning of the line (^) and look at the first character only from the set [A-Z]. If first character doesn't match , the line is skipped and the next line is examined.
- If the first character is match, the second expression (.*) matches the rest of the line until the last character which must be exclamation mark.
- The third expression examines the character at the end of the line (!).

egrep

- For example: Write a command to print lines which start with capital character and end with ! With possibility of a character following the full stop at the end.

```
$ egrep “(^[A-Z]*.!$) | (^[A-Z]*.!.$) | (^[A-Z]*.!..$)" myfile.txt
```

- It consist there group of pattern separated by alternation operator |.
- It print line which start with Upper case character and end with ! OR one character after ! (!.) OR two character after ! (!..).

fgrep

- Full Form: "Fixed-string Global Regular Expressions Print".
- fgrep which is the same as grep –F.
- It is fixed or fast grep and behaves as grep but does NOT recognize any regular expression meta-characters as being special.
- The search will complete faster because it only processes a simple string rather than a complex pattern.
- For example: if I wanted to search my .bash_profile for a literal dot (.) then using grep would be difficult because I would have to escape the dot because dot is a meta character that means 'wild-card, any single character':

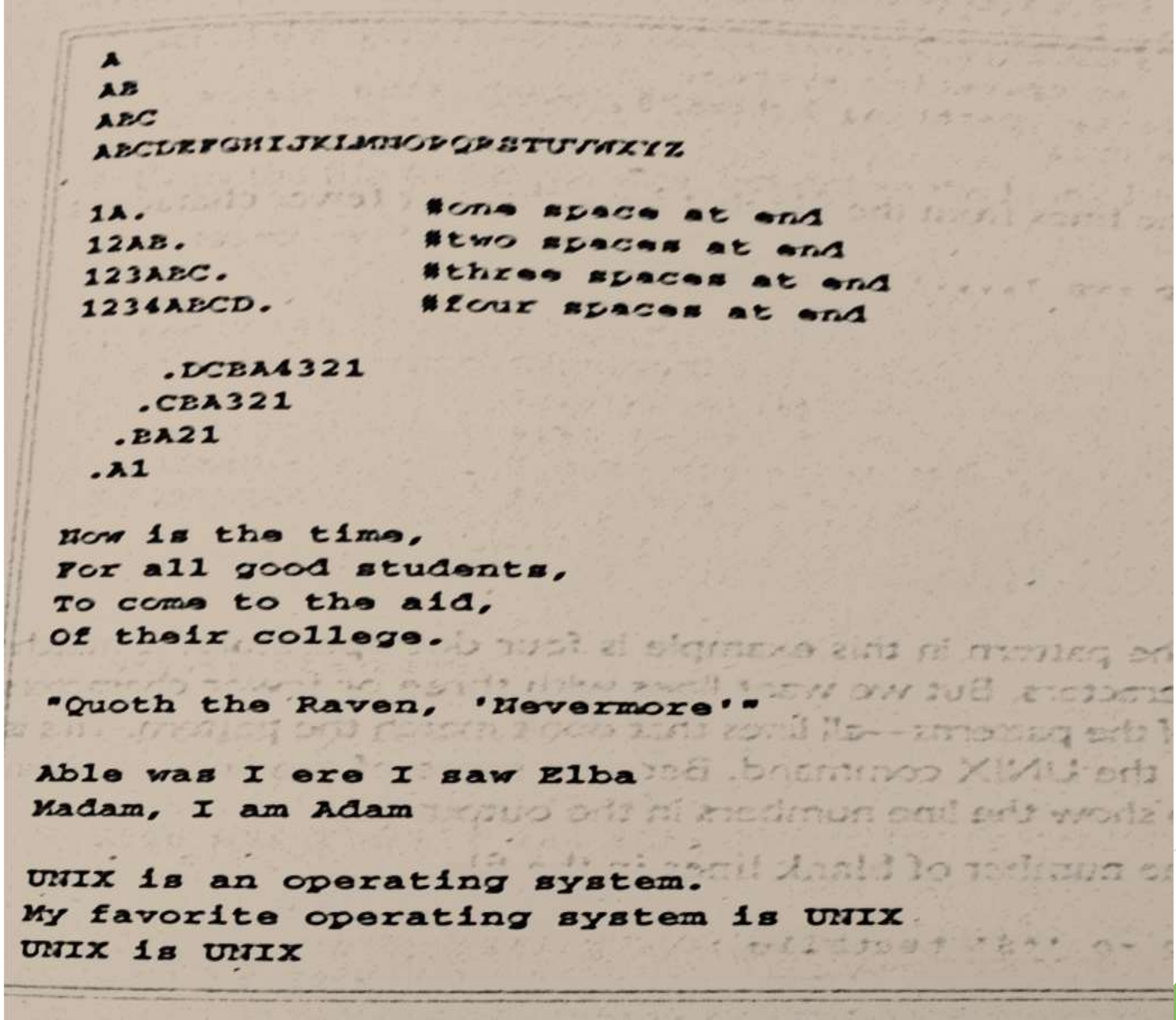
fgrep

```
$ grep "." myfile.txt
```

- The above command returns every line of myfile.txt. Do this instead:

```
$ fgrep "." myfile.txt
```

- Then only the lines that have a literal '.' in them are returned. fgrep helps us not bother escaping our meta characters.



```
A
AB
ABC
ABCDEFGHIJKLMNOPQRSTUVWXYZ

1A.                 #one space at end
12AB.               #two spaces at end
123ABC.             #three spaces at end
1234ABCD.           #four spaces at end

      .DCBA4321
     .CBA321
    .BA21
   .A1

Now is the time,
For all good students,
To come to the aid,
Of their college.

"Quoth the Raven, 'Nevermore'"

Able was I ere I saw Elba
Madam, I am Adam

UNIX is an operating system.
My favorite operating system is UNIX
UNIX is UNIX
```

Examples

Question
Select the lines from the file that have exactly three character.
Select the lines from the file that have at least three character.
Select the line from the file that have three or fewer characters.
Count the number of nonblank lines in file.
Select the line from the file that have string “UNIX”
Select the line from the file that have only string “UNIX”

Examples

Question
Select the line from the file that have the pattern UNIX at least two times.
Copy the file to the monitor, but delete the blank lines.
Select the line from the file that have at least 2 digits without any other characters in between
Select the line from the file which don’t start with A to G.
Search for a word which is having three letters in it and starts with x and ends with m.
Locate a string ‘UNIX’or ‘NETWORK’ from a file.
Locate a string ‘sengupta’ or ‘dasgupta’ from a file.
List all directory in current directory.

Examples

Answer
$ egrep ‘^…$’ file
$ egrep ‘…’ file
$ egrep –vn ‘….’ file. Here ‘ …. ‘ match all lines with at least 4 character. But we want lines with 3 or fewer character so that means we want the inverse pattern so we use “ –v “
$ egrep –c ‘.’ file
$ fgrep ‘UNIX’ file
$ egrep ‘^UNIX$’ file

Examples

Answer
$ egrep ‘UNIX*.UNIX’file.
$ egrep –v ‘^$’ file
$ egrep ‘[0-9][0-9]’ file
$ egrep ^[^A-G] file
$ grep ‘x[a-z]m’ file
$ grep –E ‘UNIX|NETWORK’ file.
$ grep –E ‘(sen|das)gupta’ file.
$ ls –al | grep ‘^d’