Overview

- Regular expressions (regex) and pattern matching/searching
- The "`grep`" utility

Regular Expression (regex)

regex is a pattern that describes a set of string

- A pattern of characters used to match the same characters in a search
  - Sometimes enclosed by two "/" , such as: /regex/
    - ex: /ring/, ~ "ring", "spring", "ringing", etc.
  - Used in many utilities: **`vi, emacs, grep, sed, awk`**, etc.
  - Supported in many languages: **`perl,python,php,java`**, etc.
- Meta characters
  - Metacharacters are characters that represent something other than themselves
  - Shell metacharacters used by the shell program
  - Regular expression metacharacters are evaluated by the program performing the regular expression matching, such as
    - **`vi, emacs,grep, perl, sed, awk, etc.`**

regex Metacharacters

- `*`

  Matches zero or more of the preceding character
- `.`

  matches any single character
- `^`

  beginning of the line or string
- `$`

  End of line
- `\`

  escape the metacharacters so it can be what it is.
- `[ ]`

  Any single character in the set, i.e. [A-Z]
- `[^]`

  Any character NOT in the set, i.e. [^A-Z], or [^ab]
- `^[^ ]`

  Lines beginning with any character not in the bracket

Examples

- **`Abc*`**
  Abc345d, AbBBB, Ab (Note: different from * in **`ls a*`**)
- **`.*`**
  Zero or more of any character, will match any patterns
- **`[a-zA-Z] <==> [[:alpha:]]`**
  Matches any letter, low or upper case
- **`[^0-9]`**
  Matches any character which is NOT a number
- **`end$`**
  Matches lines ending with “end”
- **`^start`**
  Matches lines starting with “start”
- **`\.$`**
  Matches line ending with “.”
- **`^ *$`**
  Matches blank lines

`grep`

- `grep`
  - Searches and displays lines which match a pattern in files
  - Free version: GNU grep

  `grep <option flags> pattern  filename(s)`
  - **grep does not change the content of the file(s) being searched**
- Common flags/options
  - `-n`: display the line number where the match is found
  - `-c`: display the number of lines containing the search pattern
  - `-r`: recursively read all files under each directory from current directory,
  - `-i`: ignore the case of letters
  - `-v`: prints all lines not containing the pattern (can be used to remove certain lines in a file)
  - `-l`: print out only the name of files in which the pattern is matched
  - `-w`: find pattern only if it is a word, not part of a word
  - For more options, check online with man grep

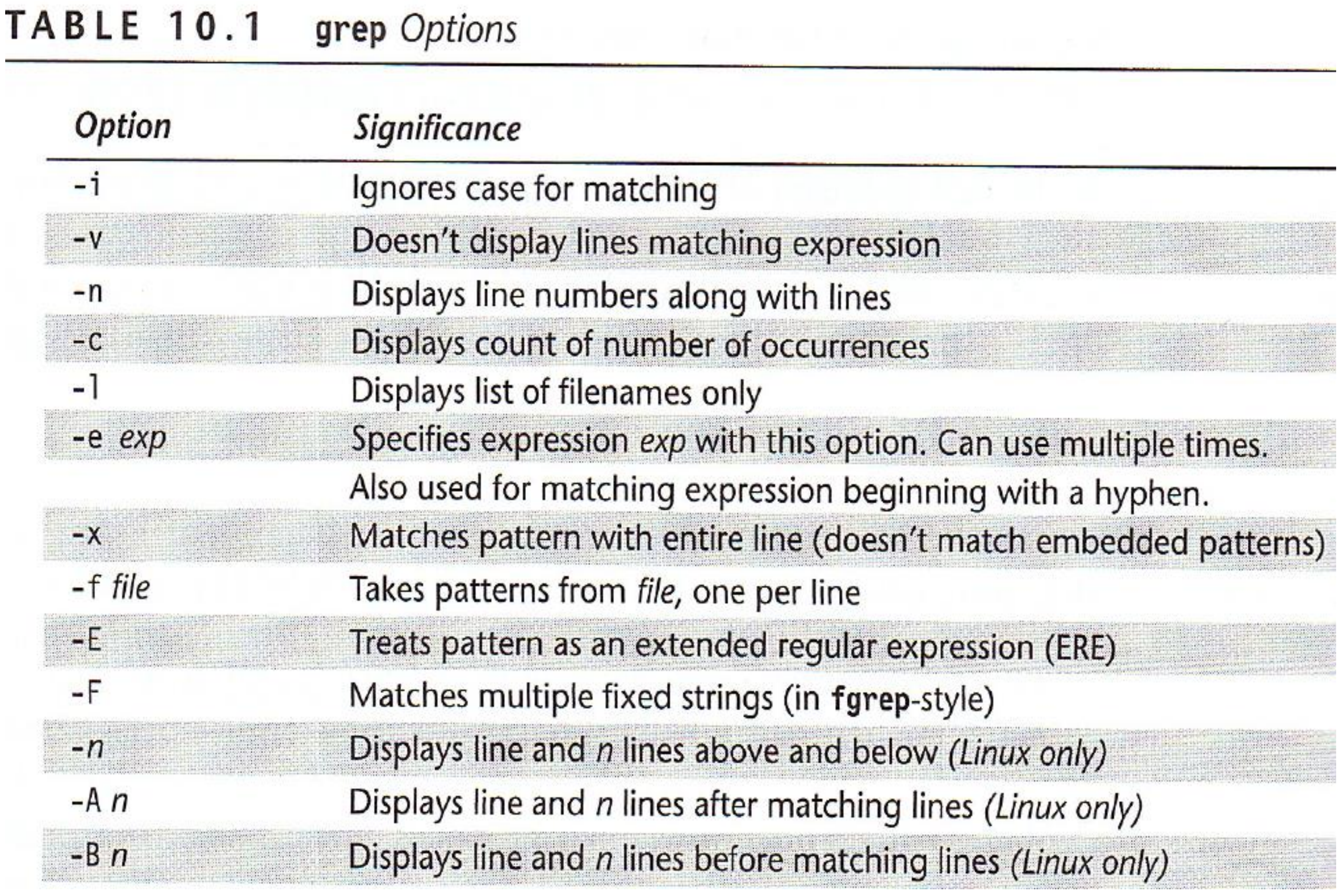

TABLE 10.1 **grep** *Options*

Option	*Significance*
-i	Ignores case for matching
-v	Doesn't display lines matching expression
-n	Displays line numbers along with lines
-c	Displays count of number of occurrences
-l	Displays list of filenames only
-e *exp*	Specifies expression *exp* with this option. Can use multiple times. Also used for matching expression beginning with a hyphen.
-x	Matches pattern with entire line (doesn't match embedded patterns)
-f *file*	Takes patterns from *file*, one per line
-E	Treats pattern as an extended regular expression (ERE)
-F	Matches multiple fixed strings (in **fgrep**-style)
-*n*	Displays line and *n* lines above and below *(Linux only)*
-A *n*	Displays line and *n* lines after matching lines *(Linux only)*
-B *n*	Displays line and *n* lines before matching lines *(Linux only)*

Examples

- `grep “NW” datafile.txt`
  - Displays lines containing “cs390” in datafile.txt
- `grep “cs390” *`
  - Displays the files and lines containing “cs390” in current directory
- `grep –r “cs390” .`
  - Search for files containing “cs390” from the current directory
- `grep “[0-9]” textfile`
  - Displays lines containing numbers in the `textfile`
- `grep “[^a-zA-Z]” datafile.txt`
  - Displays lines containing none letters in `datafile.txt`
  - Same when with “-i” option: grep -i “[^a-z]” input.txt
- `grep “^[0-9]” datafile.txt`
  - Displays lines which start with a digit
- `grep “^n” datafile.txt`
  - Display lines starting with **n**
- `grep “4$” datafile.txt`
  - Display lines ending with **4**

"grep" and pipe

- "`grep`" can take input (data stream) from a pipe
  - `ls -l | grep ^d`
    - The output of the ls command is piped to grep
    - all lines starting with "d" are printed (all the directories are displayed to the screen)
  - `who | grep hlin`
    - The name list of the current logon users are piped to grep
    - A way to check if one particular person is logon on
  - `cat file.txt | grep –w computer` ↔ `grep –w computer file.txt`

```
find . –name "*.php" | xargs grep function
```

- Search from the current directory for files ending with ".php", then the grep will search these files for lines containing "function"

```
??: find . –name *.php | grep function
```

`find` vs. `grep`

Both used to search for files but with different constraints

- **`find`**
  - search criterions on properties of the “files”, such as type, size, permissions, pattern of the file name, etc.
- **`grep`**
  - Search criterions on content of the files, looking for patterns in the content of files, you will get
    - What: Files containing the search pattern
    - Where: the lines containing the search pattern

Inverting Search with “-v”

- Show all unmatched lines
- Can be useful when you want to remove lines which contain certain pattern

```
grep –v “cs390” input.txt > input_new.txt
```

  - Display all the lines not containing “cs390”
  - Redirect the stdout to a new file “input_new.txt”
  - `input_new.txt` contains only the lines without cs390
- There are other ways to accomplish this (sed and awk)
- Questions
  - how to remove comments starting with “//” from a cpp file?
  - How to remove comments from a shell script (lines in shell script start with #)?
  - How to remove empty/blank lines from a file ?

Variants of grep: egrep & fgrep

- **`fgrep` ⇔ `grep -F`**
  - "Fixed" String grep
  - Treat all characters as literals, i.e., not metacharacters
  - Examples:
    - `fgrep "***" *.txt;   # must use DOUBLE quotes`
      - Displays all the lines containing three "*"
    - `fgrep '3.' datafile.txt`
      - Displays all the lines containing "3."
      - Will remove all the commenting lines
- **`egrep` ⇔ `grep -E`**
  - grep with extended regex, more regular expression meta-character support

New Metacharacters

- **`+`**

  Matches one or more of the character(s) preceding "+"

- **`?`** → Matches zero or one of the preceding character
- **`a|b`** → Logical "or"
- **`\(..\)`** → Matches the group
- **`x\{n\}`**
  - Number of repeat of the preceding pattern x

Examples

- `2\.?[0-9]`
  - Matches lines containing a 2 followed by zero or one period, and followed by a number
  - Matched: 2.5, 25, 29, 2.3
- `Tal+` (different from `Tal*`)
  - `Tal, Talk, Tallll`
- `Monday|Wednesday`
  - Lines containing either Monday or Wednesday
- `(no)+`
  - Matches no, nono, nononono, etc
- `(01)+`
  - Matches any binary number of 0101010101…
- `a\{5\}`
  - Matches at least 5 repeated “a”

More Examples for “egrep”

- `S(h|u)`
  - Matches Sharon, Suan
- `Sh|u`
  - Matches lines containing “Sh” or “u”
- `[A-Z]…[0-9]`
  - Prints lines containing a 5-characters set starting with a capital letter followed by three of any character, and ending with a digit number
- (Susan|Jean) Doe
  - Prints lines containing Susan Doe or Jean Doe
- `egrep –v ‘Mary’ file`
  - Prints lines NOT containing Mary

More Examples for "grep"

- `grep -n "5\.." datafile.txt`
  - Print line containing number 5 followed by a period and any single character (possible number-indexed lines)
- `grep -i "[a-z]\{9\}" datafile.txt`
  - Print lines containing at least 9 consecutive letters, lower or upper case
- `grep foo *`
  - Search all the files in the current directory, display the file names and the lines containing pattern "foo"
- `grep -r foo .`
  - Search all the files in the current directory recursively for pattern "foo"
- `grep -w "north" datafile.txt`
  - Print lines containing word north (not northwest)
- `grep "^[A-Z]" /etc/passwd`
  - print the line beginning with a capital letter
- `grep "^[A-Za-z]" filename`
  - Print lines beginning with a letter

A Note about "range" expression

- Within a bracket expression, a range expression consists of two characters separated by a hyphen.
- the range is determined using the locale's collating sequence and character set
  - For C locale, [a-d] ←→ [abcd]
  - Others: If locales sort characters in dictionary order, [a-d] ←→[aBbCcDd]
  - You can use "locale" to see how the system locale is set.
- Work around with predefined bracket expressions
  - man page for grep, the regular expression section

[:alnum:]	0-9 and a-zA-Z	`grep '[[:alnum:]]' file`
[:alpha:]	a-zA-Z	`grep '[[:alpha:]]' file`
[:digit:]	0-9	`grep '[[:digit:]]' file`
[:upper:]	A-Z	`grep '[[:upper:]]' file`
[:lower:]	a-z	`grep '[[:lower:]]' file`