CIS241

System-Level Programming and Utilities

Regular Expressions (regex)

Erik Fredericks, frederer@gvsu.edu

Fall 2025

> **Based on material provided by Erin Carrier, Austin Ferguson, and Katherine Bowers**

Basics

- **Regular expression**
  - Defines a set of one or more strings of characters
- **Simple string of characters**
  - Represents itself
- **Special/metacharacters**
  - Characters that **do not** represent themselves
- **Add in special characters**
  - Match a pattern which can represent many strings

Special characters

We'll focus on the Extended Regular Expression (ERE) syntax

Delimiter:

- Marks beginning/end of expression
- Often `/`
- Some utilities let you use other delimiters

Char	Use	Example
`\`	escape special character	`a\+b` matches "a+b"
`.`	wildcard - match any character	`.ord` matches "word", "cord"
`[]`	character class	`[bB]ob` matches "bob", "Bob"
`^`	beginning of line	`^B` matches "B" at start of line
`$`	end of line	`!$` matches "!" ending line

Char	Use	Example
`*`	match 0 or more occurrences	`bo*` matches "b", "booooo"
`?`	match 0 or 1 occurrences	`bo?` matches "b", "bo"
`+`	match 1 or more occurrences	`bo+` matches "bo", "boooo"
`{n}`	match exactly `n` occurrences	`bo{2}` matches "boo"
`{n,m}`	match between `n` and `m` occurrences	`bo{1,2}` matches "bo", "boo"
`()`	group characters	`(da)*` matches "da", "dada"
`|`	match next or previous	`hi|bye` matches "hi", "bye"

Some examples

`a` : matches the string `a`
`a+`: matches one or more `a`s
`a*`: matches zero or more `a`s

- What does `lo+l` match?
- Where does this differ from `lo*l`?

Parentheses can group characters (called a capture group)

- What does `(ab)+` match?
- Which ones of these wouldn't match? Why?
  `ab` `abab` `" "` `ba` `abababababab` `aba` `aab`

- **What does this command do?** `cat file_*`
- **What if we want that wildcard functionality in regex?**
  - `.` (a dot/period) - matches any character
- **How do we then match the same strings as the command above?**
  - `file_.*`

Applying restrictions

- **Example: what would `(b.d)+` match? `b.+d?`**
- **What if we want to restrict the wildcard to only match vowels?**
  - We use character classes `[ ]`
  - `b[aeiou]d`
  - How do `(ab)+` and `[ab]+` differ?
  - Note we can also use `tr`-like character classes:
    - `[[:digit:]] [a-z]`
      - Two sets of `[]`?
        - Outer: this is a character class!
        - Inner + colons: Use a `tr`-style set
  - Can also invert: `[^[:digit:]] [^0]`

Matching the *forbidden* characters

What does `foo.txt` match?

What if we want it to only match "`foo.txt`"?

- `foo\.txt`

What does this match?

```
a dog|cat
```

- Matches `a dog` or `cat`. Does not match `a cat` -- the `|` operator is greedy and needs parens:
- `a (dog|cat)`

Examples

What do these match?

- `The (dog|cat) ra+n away$`
- `^bee+s*`
- `[Ll][ol]{2}[ol]*`

Examples

Create a regex to match:

`ab`, `aba`, `abb`, `abba`, `abab`, `abbb`, `abaa`, and nothing else

Order of operations

	ERE Precedence (from high to low)	
1	Collation-related bracket symbols	[==] [::] [..]
2	Escaped characters	\<special character>
3	Bracket expression	[]
4	Grouping	()
5	Single-character-ERE duplication	* + ? {m,n}
6	Concatenation	
7	Anchoring	^ $
8	Alternation	|

via https://stackoverflow.com/questions/36870168/operator-precedence-in-regular-expressions/49445993#49445993

Using regexs

We are using extended regular expressions (ERE)

Use them with `grep` and `-E`:

- `grep -E "ab[ab]{2}" file.txt`

`grep` -> *global regular expression print*

`grep` returns any lines with a match

- To return just the match, add `-o`

Note: other commands use `/pattern/` to denote regex!

Exercises:

Which strings match the regex:

- `[^5][[:digit:]]+`
  - `12`, `3`, `50`, `15`, `b0`, `10000`, `1050`, `$10`, `$4.50`, `2!`

Create a regex to:

- Match `ab` `aba` `abb` `abba` `abab` `abbb` `abaa` and nothing else

What does this match?

- `[[:digit:]]{10}|\([[:digit:]]{3}\)[[:digit:]]{3}-[[:digit:]]{4}`