

## **Regular expressions**

### Perl's implementation

- ■Perl tries to match a regular expression pattern with \$ variable
- ■Enclose pattern in slashes

CSE2395/CSE3395

CSE2395/CSE3395

CSE2395/CSE3395

- // called match operator
- ▶ boolean value returned
- Pattern match may succeed on any substring
- in \$\_ that satisfies it
   first (leftmost) successful match is considered the matching substring



# **Example** # Find occurrences of "fth" in the named file. # Read lines of input into \$ , one at a time. CSE2395/CSE3395 # Check for existence of "fth" in \$\_. if (/fth/) # Success; print the line (\$ ).

## **Patterns**

### Matching characters

- ■Perl's regular expression syntax is subtly different from regular expressions in vi, grep, sed, lex, etc.
- Alphanumeric characters match themselves
  - ▶ /abc/ # matches "abc"
  - ▶ /123/ # string "123", not number 123
- Other characters require a backslash in order to match themselves
- ► /\(moo\)/ # matches "(moo)"
- /\/usr\/bin/ # matches "/usr/bin"



10

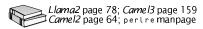
CSE2395/CSE3395

## **Patterns**

8

### Matching any character

- . (full stop) pattern matches any character except newline (\n)
- ► /d.g/ # matches "dog", "dig", "d2g", "d g"
- ▶ /..../ # matches any five characters on one line
- /..\.../ # matches any two characters, then a dot, then any two characters



## **Patterns**

### Character classes

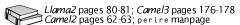
- [letters] matches any one of the enclosed letters
- ▶ /[abc]/ # matches "a", "b" or "c"
- ▶ ranges of characters can be specified
- /[a-z]/ # matches any lowercase character
- [^letters] matches any one character except those enclosed
- ▶ / $[^0-9]$ / # matches anything except digits
- Some useful character classes are predefined

11

Llama2 pages 78-79; Camel3 pages 159, 165-167 Camel2 page 64; pertre manpage

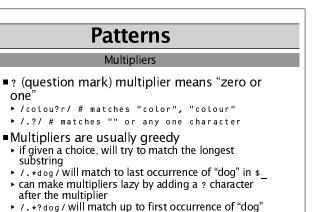
## **Patterns**

- Multipliers repeat the previous part of the pattern
- \* (asterisk) multiplier means "zero or more"
- ▶ /a\*/ # matches "", "a", "aa", "aaaaa"
- ► /[abcdr]\*/ # matches "abracadabra", "car"
- ▶ /.\*/ # matches zero or more of any character
  - will typically match the entire line
- + (plus) multiplier means "one or more"
- ► /a+/ # matches "a", "aa", "aaaaa", not ""



12

CSE2395/CSE3395



# **Patterns**

- Pattern is normally tried at all points in string
- Anchor forces match to be considered only at start or end of string
- ^ (caret) anchor forces match to start of string
- ▶ /^a/ # matches if \$\_ is "apple" but not "baa"
- s (dollar) anchor forces match to end of string
- /!\$/ # matches if \$\_ ends in a "!" character
- ► works even if newline has not been champed
- ▶ /^.\*\$/ # matches all of \$\_
- Llama2 page 83; Camel3 pages 178-180 Camel2 page 62; pertre manpage

14

CSE2395/CSE3395

## **Example** Verifying email header format # Mail headers are of the form: # Word: some text here # continuation lines are indented # Process the mail header CSE2395/CSE3395 # Stop if the line is empty (i.e., start # of string is next to end of string). last if /^\$/; # This line is valid if it starts with either: # - at least one non-space, then colon, or # - at least one space unless (/^[^\s]+:/ || /^\s+/) { print "Malformed header line:\n\$\_"; }

Llama2 pages 80-81; Camel3 pages 176-178 Camel2 pages 62-63; pertre manpage

13

CSE2395/CSE3395

**Patterns** Alternation and grouping ■ | (vertical bar) separates alternatives /cat|dog/ # matches "cat" or "dog" ► /a|b|c/ # would be better written as CSE2395/CSE3395 character class /[abc]/ ( parentheses ) used for grouping /(ab)\*/ # matches zero or more sequences
of "ab": "", "ab", "ababab" /(cat|sel)fish/ # matches "catfish" or "selfish"

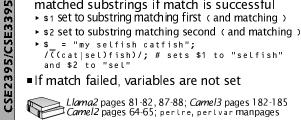
Llama2 pages 82, 84; Camel3 pages 187-188, 182-185 Camel2 pages 61, 64; pertre manpage

15 16

## Parentheses as memory Parentheses also used to remember matched ■Special variables \$1, \$2, etc. are set to matched substrings if match is successful ▶ \$1 set to substring matching first (and matching) ▶ \$2 set to substring matching second ( and matching ) \$ = "my selfish catfish"; $\sqrt{(\text{cat}|\text{sel})\text{fish})}$ /; # sets \$1 to "selfish" and \$2 to "sel"

■If match failed variables are not set

patterns



**Patterns** 

## Example Extracting email header names We want to extract the "Subject", "Date", # "From", etc. header names from an email # message, ignoring the headers' contents. # Scan mail message one line at a time. while (<>) CSE2395/CSE3395 # Finish when header is done. last if /^\$/; # Extract name of header into \$1. # This will skip continuation lines (match # will fail). if (/^([^\s]+):/) print ("Header name is \$1\n");



- Regular expressions are normally case-
- ► /a/ # will not match if \$\_ is "AARDVARK"
- ■Can make regular expressions ignore case using i modifier
- ▶ i character goes after match operator
- ► /a/i # will match "AARDVARK" now



CSE2395/CSE3395

CSE2395/CSE3395

CSE2395/CSE3395

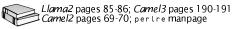
Llama2 pages 85-86; Camel3 pages 147-148 Camel2 pages 69-70; pertre manpage

19

## **Interpolation**

### Using variables in patterns

- ■Patterns are variable-interpolated like double-quoted strings before matching
- \$pattern = 'fish(es)?'; /cat\$pattern/ #
  same as matching /catfish(es)?/



CSE2395/CSE3395

20

## **Example**

#### A Perl version of grep, version 2

# perlgrep - simple replacement for grep Invoke this program as: perlgrep pattern [file ...] # Get the pattern from the command-line # Outside a function, shift's default argument # is @ARGV. \$pattern = shift; while (<>) # Print the line if it matches the pattern. o modifier is a promise that we won't change \$pattern - this allows Perl to run it faster print if /\$pattern/o;

Substitution

## Replacing a matched substring with another

- ■To replace a matched substring with a new string, use s/pattern/replacement/
- pattern is a regular expression to match to the \$ variable
- replacement is the string to replace the matching part of \$
- ► not a regular expression
- ▶ may contain \$1, \$2, etc. from matched pattern
- ■s/colou?r/hue/; # make a synonym



CSE2395/CSE3395

Llama2 pages 88-89; Camel3 pages 152-155 Camel2 pages 72-74; pertop manpage

22

21

## **Substitution**

#### Things to note

- Variables are interpolated in both parts
- ▶ s/\$regex/\$new/; # uses \$regex and \$new
- Substitution normally only occurs for the first match in a string
  - \$ = "scattered catfish"; s/cat/dog/; # \$\_
    is now "sdogtered catfish"
  - use g modifier to make substitution repeat as often as possible on the string
  - \$ = "scattered catfish"; s/cat/dog/g; #
    \$ is now "sdogtered dogfish"
  - substitution operator also takes i modifier

Llama2 page 88; Camel3 page 153 Camel2 page 72; pertop manpage

## **Example**

### Exchanging the first two words typed

# Read in a line. \$ = <STDIN>

# Switch first two non-space substrings CSE2395/CSE3395 # that are separated by space. # \S is predefined to be any non-space
# character, the opposite of \s;
# i.e., [^ \t\n\r\f] or [^\s]
s/(\S+)\s+(\S+)/\$2 \$1/;

# Print result. print;

## The = " operator

### Matching something other than \$

- Sometimes using \$ as the string to match can be awkward
- ■The = ~ (binding) operator allows the match and substitution operators to use a different
  - ▶ \$a =~ /x/ # true if variable \$a has "x" in i t
- ► \$a =~ s/x/y/; # replaces first occurrence of "x" in \$a with "y"
- ▶ \$\_ = ~ /abc/is same as just /abc/

CSE2395/CSE3395

CSE2395/CSE3395



25

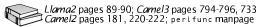
## split **and** join

#### Converting between strings and lists

- split /pattern/, string
- breaks up string into parts separated by pattern, returning the parts as a list
- ► awords = split / /, "cat and mouse"; # awords contains ("cat", "and", "mouse").
- ► awords = split /\s+/, \$line;
- join string, list

CSE2395/CSE3395

- glues the elements of list together with string, returning the result as a string
- ▶ \$joined = join " ", @words



26

# **Example**

#### Reading the password file

# Print all user names whose real name
# starts with "D".
# The /etc/passwd file is of the format: username:passwd:uid:gid:gcos:dir:shell # The real name is kept in the gcos field. # Read in the data (redirected by shell # since we can't open /etc/password yet). while (<>) # Split \$\_ into fields separated by a colon.
# Default string to split is \$\_. ainfo = split /:/; # If gcos field starts with D, print username. print " $\inf_0 01\n$ " if  $\inf_0 41 = ^/D/i$ ;

27

## **Covered today**

- Regular expressions
- Pattern matching
- ▶ /pattern/
- Substitution
- ▶ s/pattern/replace/
- Functions that use regular expressions
  - ▶ split, join

CSE2395/CSE3395

## **Going further**

### More things related to today's topic

- Not-so-regular expressions
  - far more than you ever wanted to know about
- pattern matching
   Camel3 pages 195-216
- Mastering Regular Expressions, by Jeffrey Friedl, O'Reilly 1997
- sed, awk, grep, vi, ...
- ▶ some of Unix's more powerful pattern-matching tools
- ▶ man sed, man awk,...

## **Next time**

28

### To be covered in Topic 7

- Functions
  - ▶ calling
  - accessing parameters
  - ▶ local variables

CSE2395/CSE3395

Reading: Learning Perl 2nd edition chapter 8, pages 92-100 Programming Perl 3rd edition pages 217-225 Programming Perl 2nd edition pages 111-121 perl sub manpage

30

CSE2395/CSE3395

CSE2395/CSE3395 lecture notes copyright © 2000 -2001 Deborah Pickett.
Reproduction of this presentation for nonprofit study use is permitted. All other reproduction, including for other educational courses, must be authorized in writing by the author.