CSE2395/CSE3395

Topic 6

Regular expressions

Learning Perl 2nd edition
chapter 7, pages 76-91
Programming Perl 3rd edition
pages 139-195
Programming Perl 2nd edition
pages 58-76
`perlre` manpage

CSE2395/CSE3395

Last time

Covered in Topic 5

- Hashes
- Hash variables
- Hash literals
- Functions that operate on hashes
  - `keys`, `values`, `each`, `delete`, `exists`
- Accessing Perl's environment
  - the `%ENV` variable

CSE2395/CSE3395

To be covered today

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

CSE2395/CSE3395

Detour: `grep`

Locating a string in a file

pattern (a regular expression)

```
% grep fth /usr/dict/words
fifth
fifths
lefthand
Lufthansa
softheaded
softhearted
twelfth
twelfths
```

file to search
(`/usr/dict/words` contains one word per line)

every line containing the pattern "fth" is printed

CSE2395/CSE3395

Regular expressions

An example of a successful match

each possible starting position tried in sequence

string → f i f t h s ← success

pattern → f t h

CSE2395/CSE3395

Regular expressions

Matching a string with a pattern

- Regular expressions are patterns which Perl tries to match against a string
  - patterns may be quite complex
- Pattern may match anywhere in the string
  - special symbols can anchor the search to start/end of string
- Matching returns true if pattern matches at least once in the string
- Matching returns false if pattern fails to match anywhere in the string

Llama2 pages 76-77; *Camel3* pages 139-140
Camel2 page 58

Regular expressions

Perl's implementation

- Perl tries to match a regular expression pattern with `$_` variable
- Enclose pattern in slashes
  - `/fth/`
  - `//` called match operator
  - boolean value returned
- Pattern match may succeed on any substring in `$_` that satisfies it
  - first (leftmost) successful match is considered the matching substring

Llama2 page 77; *Camel3* pages 140, 145-150
Camel2 pages 58, 69-72; `perlre`, `perlop` manpages

CSE2395/CSE3395

Example

Perl `grep fth`

```
# Find occurrences of "fth" in the named file.

# Read lines of input into $_, one at a time.
while (<>)
{
  # Check for existence of "fth" in $_.
  if (/fth/)
  {
    # Success; print the line ($_).
    print;
  }
}
```

CSE2395/CSE3395

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"`

Llama2 page 78; *Camel3* page 158
Camel2 page 59; `perlre` manpage

CSE2395/CSE3395

Patterns

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`

Llama2 page 78; *Camel3* page 159
Camel2 page 64; `perlre` manpage

CSE2395/CSE3395

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
  - `/\s/ # "space" chars: same as /[ \r\t\n\f]/`
  - `/\d/ # "digit" chars: same as /[0-9]/`

Llama2 pages 78-79; *Camel3* pages 159, 165-167
Camel2 page 64; `perlre` manpage

CSE2395/CSE3395

Patterns

Multipliers

- 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 ""`

Llama2 pages 80-81; *Camel3* pages 176-178
Camel2 pages 62-63; `perlre` manpage

CSE2395/CSE3395

Patterns

Multipliers

- ? (question mark) multiplier means "zero or one"
  - `/colou?r/ # matches "color", "colour"`
  - `/.?/ # matches "" or any one character`
- Multipliers are usually greedy
  - if given a choice, will try to match the longest substring
  - `/.*dog/` will match to last occurrence of "dog" in `$_`
  - can make multipliers lazy by adding a ? character after the multiplier
  - `/.*?dog/` will match up to first occurrence of "dog"

Llama2 pages 80-81; *Camel3* pages 176-178
Camel2 pages 62-63; `perlre` manpage

CSE2395/CSE3395

Patterns

Anchoring

- 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"`
- $ (dollar) anchor forces match to end of string
  - `/!$/ # matches if $_ ends in a "!" character`
  - works even if newline has not been `chomp`ed
  - `/^.*$/ # matches all of $_`

Llama2 page 83; *Camel3* pages 178-180
Camel2 page 62; `perlre` manpage

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
while (<>)
{
  # 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$_";
  }
}
```

CSE2395/CSE3395

Patterns

Alternation and grouping

- | (vertical bar) separates alternatives
  - `/cat|dog/ # matches "cat" or "dog"`
  - `/a|b|c/ # would be better written as 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; `perlre` manpage

CSE2395/CSE3395

Patterns

Parentheses as memory

- Parentheses also used to remember matched patterns
- 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";`
    `/(cat|sel)fish)/; # sets $1 to "selfish" and $2 to "sel"`
- If match failed, variables are not set

Llama2 pages 81-82, 87-88; *Camel3* pages 182-185
Camel2 pages 64-65; `perlre`, `perlvar` manpages

CSE2395/CSE3395

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 (<>)
{
  # 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");
  }
}
```

CSE2395/CSE3395

Case sensitivity

The `i` modifier

- Regular expressions are normally case-sensitive
  - `/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`

Llama2 pages 85-86; *Camel3* pages 147-148
Camel2 pages 69-70; `perlre` manpage

CSE2395/CSE3395

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)?/`

Llama2 pages 85-86; *Camel3* pages 190-191
Camel2 pages 69-70; `perlre` manpage

CSE2395/CSE3395

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;
}
```

CSE2395/CSE3395

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`

Llama2 pages 88-89; *Camel3* pages 152-155
Camel2 pages 72-74; `perlop` manpage

CSE2395/CSE3395

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; `perlop` manpage

CSE2395/CSE3395

Example

Exchanging the first two words typed

```
# Read in a line.
$_ = <STDIN>

# Switch first two non-space substrings
# 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;
```

CSE2395/CSE3395

The =~ operator

Matching something other than $_

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

Llama2 page 85; *Camel3* page 144-145
Camel2 page 81; `perlop` manpage

CSE2395/CSE3395

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
  - `@words = split / /, "cat and mouse"; # @words contains ("cat", "and", "mouse").`
  - `@words = split /\s+/, $line;`
- `join` string, list
  - glues the elements of list together with string, returning the result as a string
  - `$joined = join " ", @words`

Llama2 pages 89-90; *Camel3* pages 794-796, 733
Camel2 pages 181, 220-222; `perlfunc` manpage

CSE2395/CSE3395

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 $_.
  @info = split /:/;

  # If gcos field starts with D, print username.
  print "$info[0]\n" if $info[4] =~ /^D/i;
}
```

CSE2395/CSE3395

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, ...`

CSE2395/CSE3395

Next time

To be covered in Topic 7

- Functions
  - calling
  - accessing parameters
  - local variables

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

CSE2395/CSE3395

CSE2395/CSE3395