More *Perl*

CIS*2750

Advanced Programming Concepts

Regex shorthands in *Perl*

\t	• Tab character
\n	• Newline character
\s	• Matches any whitespace character
\S	• Anything not \s
\w	• [a-zA-Z0-9_] use \w+ to match a word
\W	• Anything not \w
\d	• [0-9] a digit
\D	• Anything not \d [^0-9]

Reading in & printing file(s)

```
#!/usr/bin/perl
while ( $_ = <ARGV> ) {
   print $_;
}
```

```
#!/usr/bin/perl
while ( <> ) {
   print;
}
```

invert.pl

```
#!/usr/bin/perl                  # Usage:  ./invert.pl filename
$file = shift(@ARGV);            # grab the 1st command argument (file name)
$ret = open(FH,$file);           # open that file name as file handle FH
$i = 0;
$lines[$i++] = tell(FH);         # read in all lines and store their locations
while ( <FH> ) {
   $lines[$i++] = tell(FH);
}
$i--;
while ( $i >= 0 ) {              # read in lines from the bottom to top
   seek(FH,$lines[$i--],0);
   $outp = <FH>;
   print $outp;                  # print out the line
}
```

nonum.pl

```
#!/usr/bin/perl
while ( $_ = <ARGV> ) {
   if ( /^[^0-9]+$/ ) {
         print $_;
   }
}
```

123

4

I see 5 dogs

I see five dogs

./nonum.pl testfile

????????????

nonum.pl

```
#!/usr/bin/perl
while ( $_ = <ARGV> ) {
   if ( /^[^0-9]+$/ ) {
         print $_;
   }
}
```

```
123
4
I see 5 dogs
I see five dogs

./nonum.pl testfile
I see five dogs
```

nonum1.pl

```
#!/usr/bin/perl
while ( $_ = <ARGV> ) {
   if ( /[^0-9]+/ ) {
         print $_;
   }
}
```

```
123
4
I see 5 dogs
I see five dogs

./nonum.pl testfile
??????????
```

nonum1.pl

```
#!/usr/bin/perl
while ( $_ = <ARGV> ) {
   if ( /[^0-9]+/ ) {
         print $_;
   }
}
```

```
123
4
I see 5 dogs
I see five dogs

./nonum.pl testfile
123
4
I see 5 dogs
I see five dogs
```

nonum2.pl

```
#!/usr/bin/perl
while ( $_ = <ARGV> ) {
   $var = $_;
   chop $var;
   if ( $var  =~  m/[^0-9]+/ ) {
        print $_;
   }
}
```

```
123
4
I see 5 dogs
I see five dogs

./nonum.pl testfile
??????????
```

nonum2.pl

```
#!/usr/bin/perl
while ( $_ = <ARGV> ) {
   $var = $_;
   chop $var;
   if ( $var  =~  m/[^0-9]+/ ) {
        print $_;
   }
}
```

```
123
4
I see 5 dogs
I see five dogs

./nonum.pl testfile
I see 5 dogs
I see five dogs
```

nonum3.pl

```
#!/usr/bin/perl
while ( $_ = <ARGV> ) {
   if ( /^\D+$/ ) {
         print $_;
   }
}
```

```
123
4
I see 5 dogs
I see five dogs

./nonum.pl testfile
I see five dogs
```

Regex quantfier

```
#!/usr/bin/perl
while ( $_ = <ARGV> ) {
    if ( /a{2,4}/ ) {                  # look for aa, aaa, aaaa
        $var1 = $';                    # store text after match
        chop $var1;                    # remove newline
        print "(", $&, " ", $`, " ",  $var1 , ")\n";
#     (text matched   before match   after match)
    }
    else {
        if ( /a+/ ) {                   # other amounts of  a's
          print "Any number of a's: ",$_;
        }
    }
}
```

cab

ccaabb

cccaaabbb

ccccaaaabbbb

cccccaaaaabbbbb

?????

cab

ccaabb

cccaaabbb

ccccaaaabbbb

cccccaaaaabbbbb

Any number of a’s: cab

cab

ccaabb

cccaaabbb

ccccaaaabbbb

cccccaaaaabbbbb

Any number of a's: cab

(aa cc bb)

cab

ccaabb

cccaaabbb

ccccaaaabbbb

cccccaaaaabbbbb

Any number of a's: cab

(aa cc bb)

(aaa ccc bbb)

cab

ccaabb

cccaaabbb

ccccaaaabbbb

cccccaaaaabbbbb

Any number of a's: cab

(aa cc bb)

(aaa ccc bbb)

(aaaa cccc bbbb)

cab	Any number of a's: cab
ccaabb	(aa cc bb)
cccaaabbb	(aaa ccc bbb)
ccccaaaabbbb	(aaaa cccc bbbb)
cccccaaaaabbbbb →	(aaaa ccccc abbbbb)

After-Match Special Variables

$`	Text before match
$&	Text matched
$'	Text after match

Capturing

- After a successful match, *Perl* provides variables $1, $2, $3 … which hold the text matched by their respective (parenthesized subexpressions) in the regex.
  - Subexpressions are numbered by counting open parentheses from the left starting at 1 (not 0)

```
#!/usr/bin/perl
while ( $_ = <ARGV> ) {
    if ( /(c{1,3})(a{2,4})/ ) {
        print $2, " ", $1, "\n";
    }
}
```

cab

ccaabb

cccaaabbb

ccccaaaabbbb

cccccaaaaabbbbb

?????

cab	
ccaabb	aa cc
cccaaabbb	aaa ccc
ccccaaaabbbb	aaaa ccc
cccccaaaaabbbbb	aaaa ccc

Command Line Arguments
pgm2.pl

```
#!/usr/bin/perl
# Extract the first argument from the command line
# and place it into the variable $pattern.  All
# other arguments are treated as filenames.
$pattern = shift(@ARGV);
while ( <ARGV> ) {
   if ( /$pattern/ ) {
      print $ARGV,": ",$_;
   }
}
```

Looking for a String in a File

File1

SetSize

ResetSize

SETSIZE

resetSIZE

./pgm2.pl Size File*

???????????

File2

This is a new file

and its Size is very small.

Looking for a String in a File

File1

SetSize

ResetSize

SETSIZE

resetSIZE

File2

This is a new file

and its Size is very small.

./pgm2.pl Size File*

File1: SetSize

File1: ResetSize

File2: and its Size is very small

Looking for a String in a File

File1

SetSize

ResetSize

SETSIZE

resetSIZE

./pgm2.pl “is .* small” File*

??????????

File2

This is a new file

and its Size is very small.

Looking for a String in a File

File1

SetSize

ResetSize

SETSIZE

resetSIZE

./pgm2.pl “is .* small” File*

File2: and its Size is very small.

File2

This is a new file

and its Size is very small.

```
#!/usr/bin/perl
print "Do you want Greenwich Time (yes or no)?: ";
# Get the input from STDIN
$answer = <STDIN>;
chop $answer;
if ( $answer eq "yes" ) {
    $seconds = time;
    @datetime = gmtime($seconds);
    print "Time: ",$datetime[5]," ",$datetime[4]," ",$datetime[3]," 
    ",$datetime[2]," ",$datetime[1]," ",$datetime[0],"\n";
}
else {
    $seconds = time;
    @datetime = localtime($seconds);
    print "Time: ",$datetime[5]," ",$datetime[4]," ",$datetime[3]," 
    ",$datetime[2]," ",$datetime[1]," ",$datetime[0],"\n";
}
```

More on command line args

```
# Check to see if there are any command line arguments.

# The ARGV special variable contains all the command line
# arguments after the program name.

print "The number of command line arguments: ", $#ARGV+1, "\n";

# $#ARGV returns the index number of the last item on the
# command line, i.e. if the command line was:
#   pgm.pl one two three
# then $#ARGV would contain 2.
```

```
$i = 0;
if ( $#ARGV > -1 ) {
   while ( $i <= $#ARGV ) {
      if ( -e $ARGV[$i] ) {
         if ( -d $ARGV[$i] ) {
            print $ARGV[$i]," is a directory.\n";
         }
         elsif ( -x $ARGV[$i] ) {
               print $ARGV[$i]," is an executable file.\n";
         }
         elsif ( -T $ARGV[$i] ) {
               print $ARGV[$i]," is a text file.\n";
         }
      }
      else {
         print $ARGV[$i]," is not a file.\n";
      }
      $i++;
   }
}
```

Perl's forte: Regular Expressions

- From many examples, get idea that a major use of Perl scripts is applying regexes to text files
- Hence "pcre" C library → **Perl Compatible** Regular Expressions
- Be alert to subtle differences in syntax, need for escape characters
  - pcre vs. regex library, grep vs. egrep, etc.