Introduction to Perl

MohsinA@Microsoft.Com

Windows OS and Networking

1. Using Perl on Windows

```
> dir
09/26/95  16:00              488,960 perl.exe
06/13/96  10:40                   38 test.txt

> type test.txt
Line 1
this is a test!!!
this is a test!!!
Line 4

>
> perl -ne "print if /Line/" *.txt                # Simple Grep

Line 1
Line 4

>
```

2. Substitutions in files.

```
> perl -pe "s,this,that,i" -i~ *.txt

  > type test.txt

  Line 1
  that is a test!!!
  that is a test!!!
  Line 4

  > dir t*

  06/13/96  10:40                    38 test.txt
  06/13/96  10:41                    38 test.txt~
```

3a. Putting it in all in a batch

```
> set perldir=c:\bat
> type   c:\bat\grep0.bat

  @rem = '                                # Dos and perl comment
  @goto endofperl                         # Dos command.
  ';                                      # End of perl comment

  $pat = shift || die "No pattern?\n";         # Start of perl script
  while(<>){                                   # Loop $_ = <ARGV>
     print if m/$pat/;                         # Print $_  if matched
  }

  __END__                                         # End of perl script
  :endofperl                                      # Dos label
  @perl %perldir%\%0.bat %1 %2 %3                 # Dos calls perl, script, args.
```

3b. Using the batch command

```
> grep0
No pattern?

> grep0 that test.txt
that is a test!!!
that is a test!!!

> type test.txt | grep0 that
Line 1
Line 4
```

4. Perl Data types

```
$var   - A scalar variable - string, integer, float.
        eg. $name = "$lastname, $firstname" . $id++;

@var   - An array indexed by a numbers.
        eg.     @names = ( 'john', 'mack' );
        Same as: ( $name[0], $name[1] ) = ('john','mack');

%var   - An associative array (indexed by a strings).
        eg. %tel = ( 'john', 124,
                     'mack', 2347 );

&var   - A subroutine named var.
        eg. sub double { .... }           Defn
            $y = &double( $x );            Call

*var   - Parameter passing by textual name, rather than value or ref.
```

5. Regular Expressions

```
^      = Beginning of line (anchoring).
$      = End of line (anchoring).
R?     = R or nothing.
R*     = R zero or more times.
R+     = R one or more times.
R{m,n} = R repeated m to n times.
R|S    = R or S.
(R)    = Name this match for reuse, refer to it later as $N, N is 0-9.
.      = any char (except newline (\\n)),
c      = match alphanumeric character c.
\d,    = digit [0-9],          (\\D not a digit).
\w,    = word-char [a-zA-Z_],   (\\W not ..).
\b,    = word-boundary,        (\\B not ..).
\s     = white-space char,     (\\S not ..).
\\, \[, \], \(, \), \|, \$, \^ : match literally to quoted char.
\0NNN  = Octal digit NNN.
[S]    = Any one char in string S will match.
[^S]   = Any char not in S.
```

6. Regular expression Examples

"Books?"   Matches Book and Books.
   "Book|Books"   (same)

   "(abc|xyz)\d" Will match the strings: abc0 .. abc9, xyz0 .. xyz9.

   "^A.*A$"   Matches strings that start and end with an A.

   "[+-]{2,2}"   Matches "++" or "+-" or "-+" or "--" only.

   "(\\w+).*$1   Matches if any word (named $1) is repeated in the string.

   "\s+$"   Has trailing spaces.

   "[+-]?\d+(\.\d+)?" Matches 0, 123, -1, 1.11, +3.333333,

   "\b(\w+)\s+$1" Matches "fish fish"
                    "fish   fish"
                    "fish   fishes"
            but not   "fish his".

7. Real world examples

Example:

```
        s,//.*,, if $no_cpp_comments;  # Delete cpp comments
```

Example:

```
        if(  m/\bassert\b.*[^!<>=]=[^=]/ ){     # Flag: assert(i=1)
            warn "Assert: $_ has side effects";
        }
```

Example:

```
        # Flag: for( i=1 ; i<10 ; j++ )
        if( m/                                  # Single line regexp
            for\s*\(                            # for (
            \s*([^;]*,)*([\w_]+)\s*=\s*[^;\s]+\s*;   #   i = x ;
            \s*([\w_]+)\s*[<>]=*\s*([^;\s]+)\s*;     #   i < 10 ;
            \s*([\w_]+)[+-]{2,2}\s*\)                #   j++  )
            /
           && ( ($2 ne $3 ) || ($2 ne $5)  ) )       # "i" != "i" ||
        {                                       # "i" != "j"
            warn  "Loop vars don't match.\n";
        }
```

8a. RENAME FILES - s///

```
#!/usr/local/bin/perl5
# Mosh@cse.iitb.ernet.in
$USAGE ='
 Usage: p-rename "s/regexp/regexp/ioge" FILES [or STDIN]
 Examples:
   o rename "s/\\.tex/.bak/" *.tex              | f.tex    to f.bak
   o rename "tr/A-Z/a-z/" *                     | ABC.EXE  to abc.exe
   o rename "s/0*(\\d)/.\$1/"  file0*?          | f004     to f.4
   o find . -print | rename "s/(\\d+)/\$1*5/e" | f4.tex   to f20.tex
';
$verbose=1;                                    # Turn on debugging.
$op = shift || die $USAGE;                     # One arg mandatory.
if(!@ARGV){                                    # No cmd line args?
    @ARGV = <STDIN>;                           # then use slurp stdin.
    chop(@ARGV);                               # No trailing newline.
}

# Continued
```

8b. RENAME FILES - s///

```
# continued

for(@ARGV){
     $F = $_;                                             # Save the initial name.
     eval $op;                                            # Find the new name.
     print STDERR "$F => $_ \n" if $verbose;              # Debugging?
     die $@ if $@;                                        # Failed?
     next if $F eq $_ ;                                   # No change in name.
     next if ((-e $_)  && (warn "File $_ exist\n"));      # Donot overwrite.
     rename( $F, $_ )  && (warn "Renamed $F to $_\n");  # Do it.
  }
```

9. Count words of interest - wc

```
#!/usr/local/bin/perl5 -w
 # SYNOPSIS: Count all matching words occurring in files.
 # Mosh@cse.iitb.ernet.in

  $pat = shift || die "Usage: $0 WORDPATTERN FILES\n";
  print STDERR "Counting words ~= /$pat/o \n";

  while(<>){
     @words = split(/\s+/);                          # String to array.
     foreach $word (grep( /$pat/o, @words )){        # Pick matching words
         $wordcount{ $word }++ ;                     # Keep tally count.
     }
  }

  foreach $word (sort keys %wordcount){
    printf( "%25s %04d\n",$word, $wordcount{ $word } );
  }
```

10. Generate a cross reference of perl variables

```
@ARGV || die "Generate a cross reference of perl variables.
   USAGE:    $0 perlfiles
   SYNOPSIS: Count all perl variables in files with line numbers of occurrence.\n";
 $pattern = '[\$\%\@][A-Za-z]+';        # what to count.
 while( <> ){
     s/#[^\n]*\n/\n/g;                  # No comments.
     while( s/($pattern)// ){
         $word = "\\$1";                # Need to quote dollars.
         next unless( $word =~ /.../ );  # Skip small words.
         $wordcount{ $word }++;          # How many occurrences.
         $wordline{ $word } .= "$., ";   # On which lines.
     }
 }
 # Print out the sorted words, count, line numbers.
 foreach $word (sort keys %wordcount){
        printf( "%-25s %03d %s\n",
               $word,
               $wordcount{ $word },
               $wordline{ $word }
               );
 }
```

11. Remove /* Comments */ from C source.

```
> type no-ccom.pl
$/ = undef;                          # Multi-line patterns.
$_ = <>;                             # Read in whole file!
s#/\*[^*]*\*+([^/*][^*]*\*+)*/|("(\\.|[^"\\])*"|
'(\\.|[^'\\])*'|\n+|.[^/"'\\]*)#$2#g;
print;

> type test.c
   /*** **** / ***/
   x = 1;
   y = "/* ...";
   z = '*/ ... '; /**/
   /* Testing
      "// x = y * /"
   */
> no-ccom test.c

   x = 1;
   y = "/* ...";
   z = '*/ ... ';
```

12a. Sample Debugging

Try -w flag.

```
 s#/\*                              Comment Start /*
    [^*]*                           Non stars*
    \*+                               Some stars+
    ([^/*][^*]*\*+)*                      ($1)  ...*
    /                                  /  Comment End
   |                                  OR  ($2)
   (  "(\\.|[^"\\])*"                    String  "a\"bc"
    | '(\\.|[^'\\])*'                  String  'a\'bc'
     | \n+                          Newlines
       |.[^/"'\\]*                    Any char . non-quote.
   )
 #{$&}#                              Put back what matched.
 g;                                  Global substitute.
```

12b. Sample Debugging

```
> no-ccom test.c

{    }{/*** **** / ***/}{
}{    x = 1;
    y = }{"/* ..."}{;
    z = }{'*/ ... '}{; }{/**/}{
}{    }{/* Testing
       "// x = y * /"
    */}{
}
```

Thank you

Questions?