Unix Pattern Matching

- There are inconsistencies between the ways different characters are used in different Unix applications and situations.
- In particular, characters are treated differently by the shell in filename expansion and by applications for pattern matching.
- For example, * means match 0 or more of the preceding expression in vi, gvim, sed, awk, grep and egrep, but * means match any string of characters in filename expansion

Unix Pattern Matching

- You can use single quotes (or, in some cases, double quotes) to bypass the shell and pass special characters to an application.
- Look up 'Unix in a Nutshell', section 6, Pattern Matching, for more detail
- Alternatively, look up the man pages for the applications you are using (man vi will tell you all you need for gvim).

perl - Practical Extraction and Report Language

- scanning text files
- extracting information
- printing reports
- features of C, sed, awk and sh
- When sed, awk and shell scripts are not quite enough ...
- ... and C is a bit too much
- ... perl is likely to be just right.

perl example – derived from examples by Lynda Thomas

```
manuel% more read_a_file.pl
#!/usr/local/bin/perl

print "Filename? ";
$file = <STDIN>;        # filename is on standard input
open(FILE,$file);       # open the file - FILE is a filehand
@file_content = <FILE>; # read into an array
close(FILE);
print @file_content;    # print the array
```

perl example

```
manuel% cat names
lynda dave tom rory lynda rory lynda

manuel% read_a_file.pl
Filename? names
lynda dave tom rory lynda rory lynda

manuel% ls -l read_a_file.pl
-rwxr-----   1 eds      staff
    264 Oct 13 14:33 read_a_file.pl
```


perl identifiers and types

```
manuel% cat trythis.pl
#!/usr/local/bin/perl

$scalar_string_id = "this is a string";
print $scalar_string_id . "\n";

$scalar_number_id = 7;
print "scalar_number_id = " . $scalar_number_id . "\n";

@array_id = ("peach","fig","apricot");
print @array_id[0] .  @array_id[2] . @array_id[1] . "\n\n";

%hash_id = ("peach", 2, "fig", 8, "apricot", 3);
print "We have " . @hash_id{"peach"} . " peaches\n";
print "We have " . @hash_id{"fig"} . " figs\n";
print "We have " . @hash_id{"apricot"} . " apricots\n";
```


perl identifiers and types

```
manuel% trythis.pl
this is a string
scalar_number_id = 7
peachapricotfig

We have 2 peaches
We have 8 figs
We have 3 apricots
```


perl loops – from an example by Mike Slattery and Lynda Thomas

```
manuel% cat loopy.pl
#!/usr/local/bin/perl

while (<>) {
  @words = split;
  foreach $w (@words) {
    $count{$w}++;
  }
}

@sortedkeys = sort keys(%count);

foreach $w (@sortedkeys) {
  print "$w\t$count{$w}\n";
}
```

perl loops

```
manuel% cat names
lynda dave tom rory lynda rory lynda

manuel% loopy.pl names
dave    1
lynda   3
rory    2
tom     1
```


Patterns

```
manuel% cat pattern.pl
#!/usr/local/bin/perl

while (<>) {
  @words = split;
  foreach $w (@words) {
    $count{$w}++;
  }
}

@sortedkeys = sort keys(%count);

foreach $w (@sortedkeys) {
  print "$w\t$count{$w}\n" if ($w =~ m/^l/);
}
```


Patterns

```
manuel% cat names
lynda dave tom rory lynda rory lynda marilyn

manuel% pattern.pl < names
lynda   3
```


perl - a subroutine

```
manuel% cat subroutine.pl
#!/usr/local/bin/perl

# from a program by Mike Slattery, June 1996

# swap takes a string, and replaces any words in %table by
# corresponding replacements

%table = ('i','you', 'you','i', 'my','your', 'your','my');

sub swap {
  local ($in) = @_;
  local ($w, $head, $tail, $new);

  if ($in =~/[a-z]+/) {
    $w = $&;    # the match
    $head = $`; # before the match
    $tail = $'; # after the match

    # look up $new in the table; if not found, $new = $w
    $new = $table{$w} || $w;

    # put the sentence back together
    $head.$new.&swap($tail);
  }

  else { $in }
}
```

...and the main program

```
while ($input = <>) {
   chop $input;
   $input =~ tr/A-Z/a-z/;
   print &swap($input)."\n";
}
```


Running the program

```
manuel% subroutine.pl
i hate computers
you hate computers
i like running
you like running
you are daft
i are daft
i'm daft too
you'm daft too
i love computers really
you love computers really
^D
```


perl - some url's

- http://language.perl.com/
- http://www.perl.com/pace/pub
- http://www.perl.org/