Why Use Perl Scripts?

- **Practical Extraction and Report Language**
- To automate repetitive and/or complex tasks
- Portable to all Unix systems, NT, etc.
- Why Perl?
  - Much more capable and powerful than Bourne shell
    - High level, interpreted
  - Includes features common to many programming languages
    - Low level, access to powerful built-in and system functions

Perl Strengths

- Text processing
  - Provides functionality of Unix `awk`, `sed`, `tr`, etc.
  - Regular expressions
- Report generation
- Scripting
  - Full-featured language
  - And then some: there are usually many ways to do something
- Rapid prototyping
  - Weakly typed
  - Interpreted, though very fast
  - Supports objects
  - GUI interfaces are available

Perl Weaknesses

- Ugly syntax
  - Like C++, though it cleans up a few things. For example, if statements must always have `{ }` around their bodies. That avoids surprises from statements like:
    ```perl
    if (x == 0)  // Not valid Perl syntax!
      if (y == 1)
        y1();
    else
      x1();  // x1() gets called only if x == 0 and y != 1 !!!!
    ```
- Write-Once
  - Weakly typed
  - Many ways to do something
Perl Data Types

- **Scalar**
  - Conventional scalar variables, *e.g.*,  
    
    ```perl
    $x = 33;
    $s = 'foo';
    ```

- **Array**
  - Arrays of any type, *e.g.*,  
    
    ```perl
    $x[3] = 33;  #### Set third element.
    @x = ();  #### Remove all elements.
    ```

- **Hash**
  - Hash table, *e.g.*,  
    
    ```perl
    $map{'foo'} = $x;  #### Map 'foo' to the contents of $x
    %map = ();  #### Remove all elements.
    ```

Perl Regular Expressions

- Search, *e.g.*, majoro and config filter:

  ```perl
  # taboo_headers [regexp_array] (undef) <resend>
  # If any of the headers matches one of these regexps,
  # then the message will be bounced for review.
  taboo_headers <= END
  #^From:.*postmaster/i
  #^From:.*news\.blue-sky\.com$/i
  ^^From: abuse@zNET\.com$/i
  ^Precedence: bulk$/i
  ^Subject:.*Article rejected null/i
  ^Subject:.*Automatic reply$/i
  ^Subject:.*undeliverable/i
  ^To: s*undisclosed-recipients/i
  END
  ```

Perl Example: Hostname/Address Lookup

```perl
### IP name or address?
@ip_addr = ($host = m/^[\d.]+$/i);  if (@ip_addr) {
    ($hostname, $junk, $junk, $junk, @hostaddrs) = gethostbyaddr(pack('C4', @ip_addr), &AF_INET);
} else {
    ($hostname, $junk, $junk, $junk, @hostaddrs) = gethostbyname($host);
}

foreach $addr (@hostaddrs) {
    $ip_addr = join('.', unpack(C4, $addr));
    print "$hostname ($ip_addr)\n";
}
```
Perl Foundations

- Help the programmer.
  - It's often easier to deal with weakly typed chunks of data than integers, character arrays, etc.
- Design for change.
  - For example, new keywords can be added to Perl because variables are specially identified.
- Be postmodern.
  - Different ways of expression are useful for different problems and different problem solvers.
- “Perl is composed of equal parts computer science, linguistics, art, and common sense. (Note that computer science is outnumbered three-to-one.)”

For Further Information

- man perl, especially:
  - man perldata
  - man perlfunc
  - man perldre
  - man perlop
  - man perlfaq
- O'Reilly Camel book
- comp.lang.perl newsgroup
- http://www.perl.com/perl