28

Answers to Even-numbered Exercises

2. What is the difference between an array and a hash?

   An array is an ordered list of scalars that you access using a numeric index. A hash is an unordered collection of scalars that you access using a string index.

4. Write a regular expression to match a quoted string, such as

   ```
   He said, "Go get me the wrench," but I didn't hear him.
   ```

   Two solutions are

   ```
   /"(.+?)"/
   ```

   and

   ```
   /"([^"]+)"/
   ```

6. Many configuration files contain many comments, including commented-out default configuration directives. Write a program to remove these comments from a configuration file.

```
$ cat config-squish.pl
#!/usr/bin/perl

use warnings;
use strict;

while ( <> ) {
    # Strip the trailing newline
    chomp;

    # Skip any lines that begin with a comment,
    # potentially after leading whitespace
    next if /^\s*#/;

    # Strip any trailing whitespace
    s/\s+$//;

    # If there is at least one character left on
    # the line, print it
    print "$_\n" if /./;
}
```

8. Describe a programming mistake that Perl's warnings do not report on.

   Perl does not tell you when you are using an uninitialized variable. This type of variable exists when you forget to declare the variable using **my** or when you mistype a variable name.

10. Describe the difference between quoting strings using single quotation marks and using double quotation marks.

    Just as in the shell, strings enclosed within single quotation marks do not interpolate special character sequences, whereas those enclosed within double quotation marks do.

12. Write a program that analyzes Apache logs. Display the number of bytes served by each path. Ignore unsuccessful page requests. If there are more than ten paths, display the first ten only.

    Following is a sample line from an Apache access log. The two numbers following the HTTP/1.1 are the response code and the byte count. A response code of 200 means the request was successful. A byte count of – means no data was transferred.

```
__DATA__
92.50.103.52 - - [19/Aug/2008:08:26:43 -0400] "GET /perl/automated-testing/next_active.gif
HTTP/1.1" 200 980 "http://example.com/perl/automated-testing/navigation_bar.htm"
"Mozilla/5.0 (X11; U; Linux x86_64; en-US; rv:1.8.1.6) Gecko/20061201 Firefox/3.0.0.6
(Fedora); Blazer/4.0"
```

```
$ cat log-analyzer.pl
#!/usr/bin/perl

use strict;
use warnings;

my %bytes;

while ( my $line = <> ) {
    # Search for the HTTP request, followed by response code and byte count
    # \S is "non-space characters", and \d is "digit"
    if ( $line =~ /"(\S+) (.*) \S+" (\d+) (\S+)/ ) {
        # Only use GET and POSTs.
        next unless ( $1 eq 'GET' || $1 eq 'POST' );

        # Only interested in successful requests (HTTP 200)
        next if $3 ne '200';

        # Store the path and size.
        my $path = $2;
        my $size = $4;

        # Remove any query string, such as "?parm=12", from the path.
        $path =~ s/\?.*//;

        # Add up the sizes. A size of "-" means no transfer.
        $bytes{$path} += $size if $size ne '-';
    }
    else {
        warn "Unable to find a valid request on line $.\n";
    }
}

# Get a list of all the paths that we've seen.
my @paths = keys %bytes;

# Sort the list of paths in reverse order by the number of bytes served.
@paths = reverse sort { $bytes{$a} <=> $bytes{$b} } @paths;

# Take the top 10 if there are more than 10.
@paths = @paths[0..9] if @paths > 10;

# Print the bytes used for each path.
for my $path ( @paths ) {
    printf( "%8d %s\n", $bytes{$path}, $path );
}
```