Count the frequency of words in text using Perl

hash | open | /g | split

Prev Next

Counting how many times a given (sub)string appears in a given text is a very common task where Perl is a perfect fit.

It can be counting the word density on a web page, the frequency of DNA sequences, or the number of hits on a web site that came from various IP addresses.

Impatient?

Here is an example counting the frequency of strings in a text file. The following script counts the frequency of strings separated by spaces:

```
use strict;
use warnings;

my %count;
my $file = shift or die "Usage: $0 FILE\n";
open my $fh, '<', $file or die "Could not open '$file' $!";
while (my $line = <$fh>) {
    chomp $line;
    foreach my $str (split /\s+/, $line) {
        $count{$str}++;
    }
}

foreach my $str (sort keys %count) {
    printf "%-31s %s\n", $str, $count{$str};
}
```

The task

The task basically has 3 parts:

1. Go over the (sub)strings one-by-one.
2. Store the number of occurrence of each string in a hash.
3. Generate the report.

We won't spend a lot of time on the report part. We assume that each string we want to count is less than 30 characters long so they will comfortably fit in one line of the screen together with the number of occurrence.

A hash is a perfect place to store the counters: the keys will be the string we count and the values will be the number of occurrence.

Sort the results according to the ASCII table

Given the data in a hash called `%count` we can display the results like this:

```
foreach my $word (sort keys %count) {
    printf "%-31s %s\n", $word, $count{$word};
}
```

In this case the strings are sorted according to the ASCII table.

Sort the results according to frequency

The `sort { $count{$a} <=> $count{$b} }` statement will sort the strings based on the values in ascending (growing) order. The call to `reverse` in front of the `sort` will make sure the results are in descending order.

```
foreach my $word (reverse sort { $count{$a} <=> $count{$b} } keys %count) {
    printf "%-31s %s\n", $word, $count{$word};
}
```

Store the number of occurrence of each string

Now that we know how we are going to print the results, let's assume the words are given in an array.

```
use strict;
use warnings;

my @strings = ('hello', 'world', 'hello', 'Perl');

my %count;

foreach my $str (@strings) {
    $count{$str}++;
}
```

```
11.
    foreach my $str (sort keys %count) {
        printf "%-31s %s\n", $str, $count{$str};
    }
```

We iterate over the elements of the array using `foreach` and increment the counter of the current string. At first the hash is going to be empty. When we first encounter the string 'hello' `$count{$str}` will not yet exists. Luckily if we access a hash element where the key does not exists yet, Perl will return undef.

Then we are trying to increment that undef using `++`. In numerical operations, such as `++` and undef will behave as if it was a `0`. Though in most cases the operation on undef will generate a Use of uninitialized value warning, but