Day 16: Script Development

Suggested Reading:

The Story of Mel *(slide 5)*

A good book (not about Perl)

Homework Review

Priorities

Efficiency

Correctness

Clarity

The Story of Mel

http://rixstep.com/2/2/20071015,01.shtml

Efficient (?) but Unclear

```
opendir $dir_fh, $cwd;
my @dir_stack = readdir $dir_fh;
@dir_stack = grep { !/^\.{1,2}$/ &&
                    {$_ = "$cwd/$_"} }
                    @dir_stack;
push @stack, reverse @dir_stack;
```

http://stackoverflow.com/questions/8479919/perl-using-loop-or-map-grep

```
opendir $dir_fh, $cwd;
foreach my $entry (readdir $dir_fh) {
    next if $entry =~ /^\.\.?$/;
    push @stack, "$cwd/$entry";   # or unshift
}
```

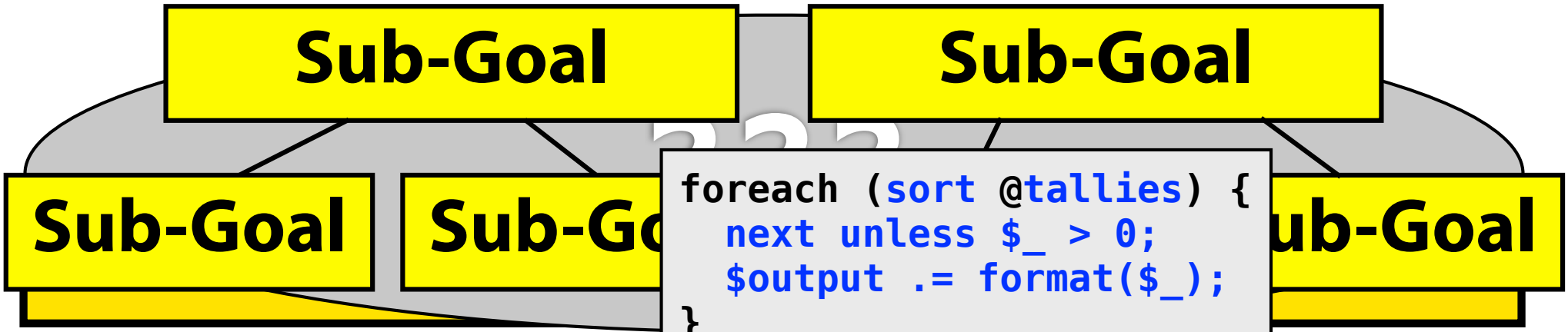
Inefficient and Unclear

```
my @words;
foreach $line (@wordlist) {
    chomp($line);
    $line = lc($line);
    my $found = 0;
    foreach my $word_ref (@words) {
        if ($word_ref->[0] eq $line) {
            $found = 1;
            $word_ref->[1]++;
        }
    }
    if (not $found) {
        push(@words, [$line, 0]);
    }
}
```

Good Design is Often Efficient and Clear

```
my %words;
foreach $line (@wordlist) {
    chomp($line);
    $words{lc($line)} += 1;
}
```

Script Development




```
foreach (sort @tallies) {
  next unless $_ > 0;
  $output .= format($_);
}
```

```
if (exists $hash{$key}) { … }
```

```
strftime(…)
```

```
foreach (@list) {
    … $_ … ;
}
```

```
unless (system(…)) {
    die "… $!\n";
}
```

```
$string =~ s/…/…/ig;
```

```
$count += 1;
```

```
open(my $f, '<', $filename)
    or die "...: $!\n";
my @lines = <$f>;
close($f);
```

```
split(/,/, @lines)
```

```
sub write_file {}
```

(Design, Code, Test)+

Refactor Mercilessly

HW #14: Phase I

```
# Read and process forecasts
# Read and process observations
# Foreach forecast, compare to observations
# Print report
```

HW #14: Phase II

```
# Read and process forecasts
open(my $forecast_fh, '<', $FORECAST_FILE)
    or die "open $FORECAST_FILE: $!\n";
while (my $forecast_line = <$forecast_fh>) {
    chomp($forecast_line);
    print "$forecast_line\n";
}
close $forecast_fh;

# Read and process observations
# Foreach forecast, compare to observations
# Print report
```

HW #14: Phase III

```
sub read_file() { ... }
# Read and process forecasts
my @f_lines = read_file($FORECAST_FILE)
    or die "open $FORECAST_FILE: $!\n";
foreach my $forecast_line (@f_lines) {
    chomp($forecast_line);
    print "$forecast_line\n";
}

# Read and process observations
# Foreach forecast, compare to observations
# Print report
```

HW #14: Phase IV

```
sub read_file() { ... }
# Read and process forecasts
my @f_lines = read_file($FORECAST_FILE)
    or die "open $FORECAST_FILE: $!\n";
my %forecasts;
foreach my $forecast_line (@f_lines) {
    chomp($forecast_line);
    # print "$forecast_line\n";
    my ($date, $time, $high, $low) =
        split("\t", $forecast_line);
    $forecasts{$date} = [$high, $low];
}
print Dumper(\%forecasts);
```

Questions?

Thank You!