Day 12: Recipes I

Dates, Times, Writing Files

Suggested Reading:
Perl Cookbook (2nd Ed.)

Chapter 3: Dates and Times
Chapter 7: File Access (esp. 7.11, 7.19)

Homework Review

Homework Preview

Forecast Sample

```
...
<H1>Madison Forecast</H1>\n
Local Madison Forecast\n
635 AM CDT THU JUL 26 2012\n
<br><br><font size=+1><B>TODAY...</B></font>PARTLY
SUNNY. A 20 PERCENT CHANCE OF THUNDERSTORMS IN\n
THE AFTERNOON. HIGHS IN THE UPPER 80S. NORTHWEST
WINDS 5 TO\n
10 MPH.\n
<br><br><font size=+1><B>TONIGHT...</B>
</font>PARTLY CLOUDY. CHANCE OF THUNDERSTORMS
THROUGH AROUND\n
MIDNIGHT...THEN SLIGHT CHANCE OF THUNDERSTORMS
AFTER MIDNIGHT.\n
LOWS IN THE MID 60S. NORTHWEST...\n

2012-07-26   06:35   UPPER 80S   LOWER 60S
```

Dates and Times

What Is So Hard About This?

Dates

- Different calendars
- Historical calendar changes
- Y2K, 2038 January 19

Times

- Coordinated Universal Time (UTC) vs. time zones
- Daylight saving time
- Leap years
- Leap seconds
- Indiana *(http://en.wikipedia.org/wiki/Time_in_Indiana)*

Unix/POSIX/Epoch Time

Seconds since **1970 January 01 @ 0:00** (UTC)

(more or less)

Remaining challenge

Unix time <–> Other time formats

Standard Date/Time Functions

`localtime`	get local YMDHMS from Unix time
`gmtime`	get UTC YMDHMS from Unix time
`Time::Local::timelocal`	create Unix time from local YMDHMS
`Time::Local::timegm`	create Unix time from UTC YMDHMS
`POSIX::mktime`	create Unix time from local YMDHMS
`POSIX::strftime`	format a Unix time

Caution: Read perldoc pages!!!

```
my ($sec, $min, $hour, $mday, $mon, $year,
  $wday, $yday, $isdst) = localtime(time);

my $real_year = 1900 + $year;
```

Parsing Dates and Times

- Use regular expressions

```
if (m,(\d{1,2})/(\d{1,2})/(\d{4}),) {
    my $year = $3 - 1900;
    my $month = $1 - 1;
    my $mday = $2;
}

my $unixtime =
  timelocal(0, 0, 0, $mday, $month, $year);
```

- Use CPAN's **Date::Manip** or **Date::Manip::Date**

Time Interval Calculations

- Use Unix time
  - Convert to Unix time
  - Do math in seconds
  - Convert to meaningful output manually

```
use POSIX qw/floor/;
use Time::Local qw/timelocal/;
my $start = timelocal(0, 0, 11, 18, 6, 112);
# Calculate interval since course started
my $interval = time() - $start;
my $minutes = floor($interval / 60);
my $seconds = $interval - ($minutes * 60);
```

- Use CPAN's **Date::Calc**

Timing Events

- Unix times: second-level resolution

```
my $start = time();
sleep(rand(10));
my $end = time();
printf "Slept %f seconds\n", $end - $start;
```

- **Time::HiRes**: much higher resolution

```
use Time::HiRes qw/gettimeofday/;
my $start = gettimeofday();
sleep(rand(10));
my $end = gettimeofday();
printf "Slept %f seconds\n", $end - $start;
```

Writing Files

Seriously?

What is so hard about writing a file?

>_<

Writing Files **Robustly**

aka

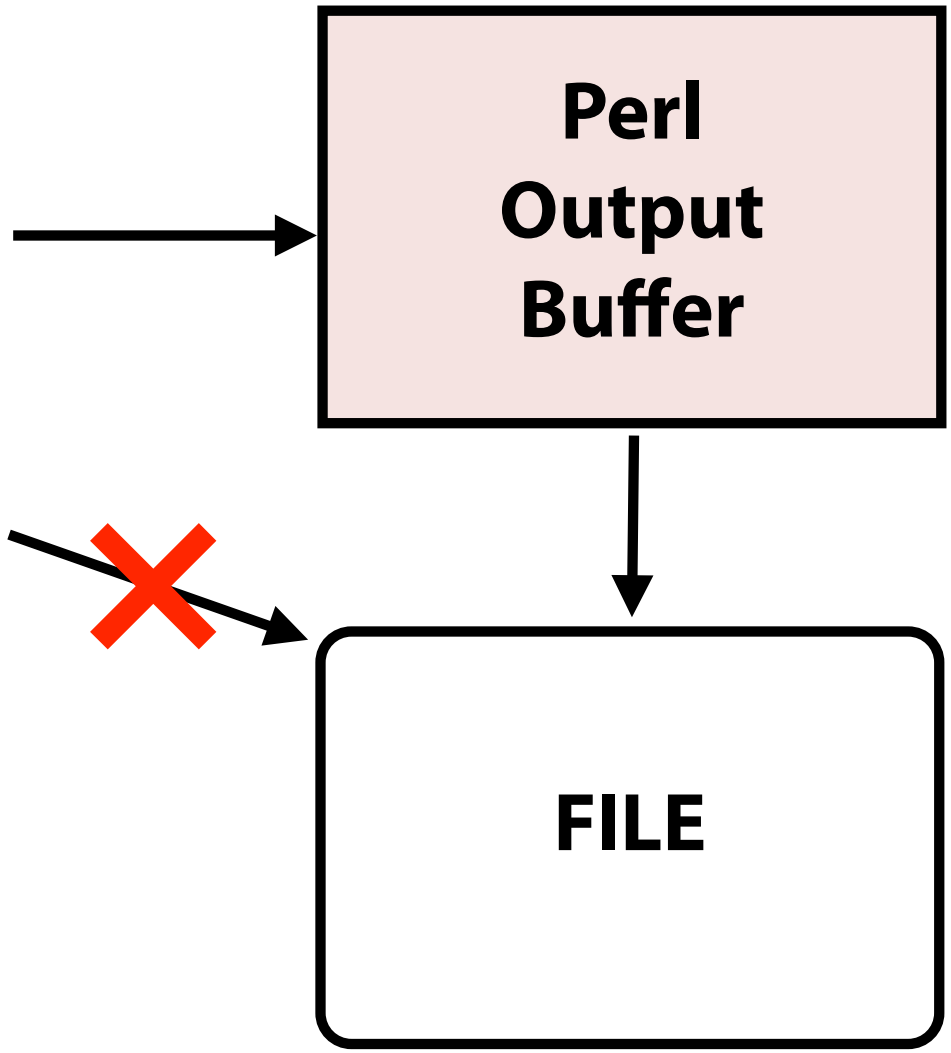
What if the power goes off in the middle of a file write?

Writing Files: The Problem

```
open(OUT, '>', $filename) or die("…");
print OUT "Header\n";
# Do some long calculation => @things [1000ms]
foreach my $thing (@things) {
  # Format $thing => $display_thing [5ms each]
  print OUT $display_thing;
}
close(OUT);
```

Perl Output Buffers

```
open(OUT, '>', $filename) ...
print OUT "first line\n";
sleep(30);
foreach my $thing (@things) {
  sleep(10);
  print OUT calculate($thing);
}
close(OUT);
```



Flushing Your … Buffers

- Set **$|** (dollar-pipe) to true
- Affects ***all*** output buffers
- Can significantly affect performance
- Not a general solution, but sometimes useful

```
#!/usr/bin/perl
use strict; use warnings;

$| = $ARGV[0];                # try 0, then try 1

print "Start of output... ";
sleep(2);
print "and now we are done!\n";
```

Atomic File Writes

Key idea: Write to separate file, move into place

```
sub write_file {
  my ($filename, $contents) = @_;
  open(NEW, '>', "$filename.NEW")     or return;
  print NEW $contents                 or return;
  close(NEW)                          or return;
  rename($filename, "$filename.BAK") or return;
  rename("$filename.NEW", $filename) or return;
  return 1;
}
```

Better Temporary Files

Use `File::Temp` to open file and give name

```
use File::Temp qw/tempfile/;
my $temp_fh = tempfile();

my ($fh, $temp_filename) = tempfile();
print $fh $contents;
close($fh);

rename($filename, "$filename.BAK");
rename($temp_filename, $filename);
```

Homework

Forecast Sample

```
...
<H1>Madison Forecast</H1>\n
Local Madison Forecast\n
635 AM CDT THU JUL 26 2012\n
<br><br><font size=+1><B>TODAY...</B></font>PARTLY
SUNNY. A 20 PERCENT CHANCE OF THUNDERSTORMS IN\n
THE AFTERNOON. HIGHS IN THE UPPER 80S. NORTHWEST
WINDS 5 TO\n
10 MPH.\n
<br><br><font size=+1><B>TONIGHT...</B>
</font>PARTLY CLOUDY. CHANCE OF THUNDERSTORMS
THROUGH AROUND\n
MIDNIGHT...THEN SLIGHT CHANCE OF THUNDERSTORMS
AFTER MIDNIGHT.\n
LOWS IN THE MID 60S. NORTHWEST...\n

2012-07-26   06:35   UPPER 80S   LOWER 60S
```

Weather Analysis, Part I

- Download current forecast
- Parse forecast timestamp and ***convert to Unix time***
- Parse temperature forecasts
- Save the data
  - ***Use safe file-write pattern with tempfile()***
  - Filename contains date of forecast timestamp
  - Record forecast timestamp and high/low predictions
  - If script is run twice in one day, overwrite