

Notes from Well House Consultants

These notes are written by Well House Consultants and distributed under their Open Training Notes License. If a copy of this license is not supplied at the end of these notes, please visit

*<http://www.wellho.net/net/whcotnl.html>
for details.*

1.1 Well House Consultants

Well House Consultants provides niche training, primarily but not exclusively in Open Source programming languages. We offer public courses at our training centre and private courses at your offices. We also make some of our training notes available under our "Open Training Notes" license, such as we're doing in this document here.

1.2 Open Training Notes License

With an "Open Training Notes License", for which we make no charge, you're allowed to print, use and distribute these notes provided that you retain the complete and unaltered license agreement with them, including our copyright statement. This means that you can learn from the notes, and have others learn from them too.

You are NOT allowed to charge (directly or indirectly) for the copying or distribution of these notes, nor are you allowed to charge for presentations making any use of them.

1.3 Courses presented by the author

If you would like us to attend a course (Java, Perl, Python, PHP, Tcl/Tk, MySQL or Linux) presented by the author of these notes, please see our public course schedule at

<http://www.wellho.net/course/index.html>

If you have a group of 4 or more trainees who require the same course at the same time, it will cost you less to have us run a private course for you. Please visit our onsite training page at

<http://www.wellho.net/course/otc.html>

which will give you details and costing information

1.4 Contact Details

Well House Consultants may be found online at

<http://www.wellho.net>

graham@wellho.net

technical contact

lisa@wellho.net

administration contact

Our full postal address is

404 The Spa
Melksham
Wiltshire
UK SN12 6QL

Phone +44 (0) 1225 708225

Fax +44 (0) 1225 707126

Handling Dates and Time

Was 8 p.m. on 28th February in Los Angeles before or after 8 a.m. in Tokyo on 1st March? Perl has a very neat trick to handling dates and times, meaning that such questions can be easily resolved. We'll show you that trick, and we'll show you the various functions that you can use to handle dates and times in Perl.

<i>So far</i>	4
<i>How Perl handles dates and times.</i>	5
<i>Convertors</i>	6
<i>Handling centuries.</i>	8
<i>Elapsed time sleep</i>	8
<i>Summary</i>	9

"Which is the most recent file of"

"How long has this program been running?"

"What will the date be three weeks from today?"

Common questions handled in Perl requiring comparisons of date and time. Hardly an easy job with the tools you've seen thus far. Of course, Perl can make it easy!

2.1 So far

File status operators

You've written

```
$modded = -M "subtwo"
```

to tell you how long ago the file "subtwo" was last modified

```
#!/usr/bin/perl
# dt1 - file last modified

print "file: ";
chop ($name = <STDIN>);

$modded = -M $name;

print "$name was last altered
$modded days ago\n";
```

```
seal% dt1
file: subtwo
subtwo was last altered 9.450798611111111 days ago
seal%
```

Figure 1 Running Perl program "dt1"

That figure is reported in decimal days.

The following options are also available:

- A days since file was last accessed
- C days since header last changed¹

stat on a file

If you **stat** a file, you can get back the same information, but in seconds from Midnight, 1st January 1970 (this time is also known as the epoch)

```
#!/usr/bin/perl
# dt2 - file last modified

print "file: ";
chop ($name = <STDIN>);

@finfo = stat($name);

print "$name was last altered at $finfo[9]\n";
```

```
seal% dt2
file: subtwo
subtwo was last altered at 1053071981
seal%
```

Figure 2 Running Perl program "dt2"

¹ not WIN32

Via system commands

You've learnt how you can get the current date and time by using system commands. You've also been told not to do this if you want portable code!

```
#!/usr/bin/perl
# dt3 - current date and time

$current = `date`;
print "Current date and time is $current";

$day = `date +%A`;
print "Today is $day";
```

Figure 3 Running Perl program "dt3"

```
seal% dt3
Current date and time is Sat Sep 20 06:16:54 BST 2003
Today is Saturday
seal%
```

2.2 How Perl handles dates and times

You've already seen a range of units and facilities ... and there's more to come. But if you work with dates and times, you'll want to convert them to more convenient units than days, months, years, hours, minutes and seconds. In other words, to easily answer a question like:

"What will the date and time be three weeks, two days and seven hours from today?"

It's easy!

- Get the current date and time in whatever units
- Convert to seconds from 1.1.1970¹
- Do your arithmetic
- Convert back to whatever units you want to report in

In other words, all dates and times are reduced to seconds from the epoch, and converters are provided to work each way.

Amazingly, you can also work in seconds before the epoch, so the scheme is valid from 1904 through to 2038 before nasty things start happening.

Other date information available

```
stat2
time()           current time and date
$^T (or $BASETIME) time the script started
```

Finally, you can set the times that a list of files were last accessed and the time they were last modified using the **utime** function.

```
utime ($acc,$mod,@files);
```

In Linux and Unix terms, this is the equivalent of the "touch" command although empty files are not created as you give a new name.

¹ for the Macintosh, operating systems prior to OSX, the time in seconds is from 1.1.1904; with OSX, time is compatible with other systems - i.e. from 1.1.1970

² covered earlier

2.3 Convertors

Convert from epoch seconds into "human readable" form:

gmtime epoch seconds to Greenwich Mean Time
localtime epoch seconds to local time
 Let's look forward six weeks and 100 weeks:

```
#!/usr/bin/perl
# dt4 - look forward weeks

print "How many weeks? ";
chop ($weeks = <STDIN>);

$now = time();
$forward = $now + $weeks * (60 * 60 * 24 * 7) ;

treport("Today it is ", $now);
treport("In $weeks weeks it will be ", $forward);

#####

sub treport {

my ($text, $timing) = @_;

($sec, $min, $hour,          # second, minute, hour
 $mday, $month, $year,      # day of month, month
 # (0-11), year
 $yday, $yday, $dst) =     # day of week, day of
 # year, daylight
 localtime($timing);

printf ("%s %d/%d/%d\n", $text, $mday, $month+1, $year+1900);
}
```

Figure 4 Running Perl program "dt4"

```
seal% dt4
How many weeks?20
Today it is 16/5/2003
In 20 weeks it will be 3/10/2003
seal% dt4
How many weeks?-200
Today it is 16/5/2003
In -200 weeks it will be 16/7/1999
seal%
```

You'll notice:

- Months come back as 0 to 11, not 1 to 12
 - Years are returned from 1900. Watch years after 1999!
 - We can tell the day of the week
 - We can tell whether daylight saving is in effect
- In a scalar context, **localtime** and **gmtime** will return a string containing the date in a nice format

Convert from human readable form to epoch seconds

Not such a common requirement, so you need to use a module which is supplied

as standard with the Perl distribution

```
use Time::Local;
```

and you can then use function

```
timelocal to convert local time to seconds
```

```
timegm to convert GM Time to seconds
```

```
#!/usr/bin/perl
# dt5 - compare a stated date and time with now!

use Time::Local;
print "When do you want? ";
chop ($timeline = <STDIN>);
($day,$month,$year,$hour,$min) =
($timeline =~
  /\^s*(\d+)\./(\d+)\./(\d+)\s+ # date
  (\d+):(\d+)\s*/x); # time
$now = time();
$then = timegm(0,$min,$hour,$day,$month-1,$year);

$diff = $then -$now;
@tsplit = splittime($diff,60,60,24,7);
treport("It is ",$now);

print "You looked ",($diff>0)?"forward ":
"backward ",
"$tsplit[4] weeks ",
"$tsplit[3] days ",
"$tsplit[2] hours ",
"$tsplit[1] minutes \n";
#####
sub treport {
my ($text,$timing) = @_;

($sec,$min,$hour, # second, minute, hour
 $mday,$month,$year, # day of month, month
 # (0-11), year
 $yday,$yday,$dst) = # day of week, day of
 # year, daylight
gmtime($timing);

printf (
 "%s %d/%d/%d %02d:%02d\n"
 , $text, $mday, $month+1, $year+1900,
 $hour, $min);
}
#####
sub splittime {
my ($val,@list) = @_;

$val = abs($val);
foreach $factor (@list){
push @rv,$val*$factor;
$val/= $factor;
}
push @rv,int($val);
return @rv;
}
}
```

Figure 5 Running Perl program "dt5"

```

seal% dt5
When do you want?28/12/79 06:15
It is 16/5/2003 08:04
You looked back 1220 weeks 0 days 1 hours 49 minutes
seal% dt5
When do you want?09/08/03 11:30
It is 16/5/2003 08:05
You looked forward 12 weeks 1 days 3 hours 24 minutes
seal%

```

2.4 Handling centuries

Whilst it is easily possible to write code that will work with old data for years starting 19xx, and also for current 20xx data, you need to take care as you program.

In particular, please note:

- **gmtime** and **localtime** return the year since 1900 so that you'll get back the number "103" for the year 2003, for example. If you want a two-digit year, you could format it using:

```
$yr = sprintf("%02d", $yrfromgmtime % 100);
```

or for a four-digit year:

```
$yr = sprintf("%4d", $yrfromgmtime + 1900);
```

- **timegm** and **timelocal** take a two-digit year code. If the number you pass in is 00 to 37, it's assumed you mean 2000 to 2037. Enter a number from 39 to 99, and Perl will use 1939 to 1999.

2.5 Elapsed time sleep

All of the functions and codes above work with dates and time. What if you want to refer to a number of seconds within your program?

```
sleep(10)    sleep 10 seconds
```

```
sleep()     sleep "forever"
```

What's the point in sleeping forever? Until an external signal is received from another process. We'll look at that more on our advanced courses.

alarm

Sets an alarm clock, so that a signal goes off after a given number of seconds. You'll study the full mechanism on the Perl Advanced course, but here's a taster if you just have a simple requirement:

```

#!/usr/bin/perl
# chivvy - hurry the user up!

@waken = (20,15,10,10,5);

$SIG{"ALRM"} = "comeon";

print "Enter your name: ";
until ($yousaid or not $waken[$kp])
{
    alarm ($waken[$kp]);
    $yousaid = <STDIN>;
    $kp++;
}

$yousaid ?

```

```

print "You entered $yousaid":
print "You failed to respond in time\n";
#####
sub comeon {
$at += $waken[$kp];
print "\n",60-$at," seconds left ... ";
}

```

Figure 6 Running Perl program "chivvy"

```

seal% chivvy
Enter your name:
40 seconds left ...
25 seconds left ...
15 seconds left ...
5 seconds left ...
0 seconds left ... You failed to respond in time
seal% chivvy
Enter your name:Graham Ellis
You entered Graham Ellis
seal% chivvy
Enter your name:
40 seconds left .Graham Ellis
You entered Graham Ellis
seal%

```

- the `%SIG` is an array of signals, into which is placed the name of subroutines to be called when a particular signal is received – in this case, `ALRM`
- `alarm` is used to set an interval timer. Perl will break out of any waiting statement when the timer goes off, firstly performing the "comeon" subroutine, then continuing with the next statement.
- the `<>` operator knows that it hasn't returned any of your typing if a signal occurred, so the text remains in the input buffer.

2.6 Summary

All date mathematics are handled in epoch seconds. Such numbers are given directly to you by the `stat` function, by the `time` function, and in the `$_B` variable.

You can convert from epoch seconds to date and time using `localtime` or `gmtime`, and from date and time to epoch seconds using `timelocal` or `timegm` from the `Time::Local` module.

Epoch seconds work fine from 1904 to 2038, but you must take care with conversions, etc, to ensure that the programs you write are cross-century compliant.

An alarm clock can be set up to signal back to you after a certain interval, allowing your process to sleep or wait. This also allows you to time out users who are slow in replying to a prompt.

Exercise

Write a program to print out the time elapsed between each web site access from the machine "catfish".
(You might like to start from our example program "wrt2", or your equivalent.)

Our example answer is `webgap`

Sample

```
graham@otter:~/profile/answers_pp> webgap
28/Aug/1998:09:47:39
seconds gap: 655
0 days 0 hours 10 minutes 55 seconds
28/Aug/1998:09:58:34
seconds gap: 67
0 days 0 hours 1 minutes 7 seconds
28/Aug/1998:09:59:41
seconds gap: 9073563
105 days 0 hours 26 minutes 3 seconds << Is this correct? Yes!!! Why?
11/Dec/1998:09:25:44
seconds gap: 101
0 days 0 hours 1 minutes 41 seconds
11/Dec/1998:09:27:25
graham@otter:~/profile/answers_pp>
```

License

*These notes are distributed under the **Well House Consultants Open Training Notes License**. Basically, if you distribute it and use it for free, we'll let you have it for free. If you charge for its distribution of use, we'll charge.*

3.1 Open Training Notes License

Training notes distributed under the **Well House Consultants Open Training Notes License** (WHCOTNL) may be reproduced for any purpose PROVIDE THAT:

- This License statement is retained, unaltered (save for additions to the change log) and complete.
- No charge is made for the distribution, nor for the use or application thereof. This means that you can use them to run training sessions or as support material for those sessions, but you cannot then make a charge for those training sessions.
- Alterations to the content of the document are clearly marked as being such, and a log of amendments is added below this notice.
- These notes are provided "as is" with no warranty of fitness for purpose. Whilst every attempt has been made to ensure their accuracy, no liability can be accepted for any errors of the consequences thereof.

Copyright is retained by Well House Consultants Ltd, of 404, The Spa, Melksham, Wiltshire, UK, SN12 6QL - phone number +44 (1) 1225 708225. Email contact - Graham Ellis (graham@wellho.net).

Please send any amendments and corrections to these notes to the Copyright holder - under the spirit of the Open Distribution license, we will incorporate suitable changes into future releases for the use of the community.

If you are charged for this material, or for presentation of a course (Other than by Well House Consultants) using this material, please let us know. It is a violation of the license under which this notes are distributed for such a charge to be made, except by the Copyright Holder.

If you would like Well House Consultants to use this material to present a training course for your organisation, or if you wish to attend a public course is one is available, please contact us or see our web site - <http://www.wellho.net> - for further details.

Change log
Original Version, Well House Consultants, 2004

Updated by: _____ on _____
Updated by: _____ on _____
Updated by: _____ on _____
Updated by: _____ on _____
Updated by: _____ on _____
Updated by: _____ on _____

License Ends.