

Beginning Perl Programming (X52.9543.001)**Handout 5: June 22, 2000*****An Interesting Output Format***

Here's some output from Homework #3:

```

35%                <----- 45% ----->                20%
-----|-----|-----
                $17160.33                $25010.85

```

A Bizarre Bug

This is a “stylized” example of a problem from Homework #3:

```

#!/App/Perl/bin/perl

&s(a, b, c);
&t(d, e, f);

sub s {
    my $s1 = @_;
    print "s1=$s1\n";
}

sub t {
    my ($t1) = @_;
    print "t1=$t1\n";
}

```

What will this program print?

An Interesting Pattern Match

Here's code from Homework #2.

```

($issueMonth =~ /jan/i) && ($issueMonth = "01");
($issueMonth =~ /feb/i) && ($issueMonth = "02");
...
($issueMonth =~ /dec/i) && ($issueMonth = "12");

```

What does it do? Does it work? Could it be better?

((inefficiency, loop, what if “mar” is a substr of a later month??))

hashcall.pl

```
#!/App/Perl/bin/perl -w
#
# hashcall.pl
#
# Demo of using a Perl hash in a subroutine call to implement named parameters.

print join("\t", qw(sec min hour mday mon year wday yday isdst)), "\n";

# Test calls to fetch local time

print "\nloctime(); returns ", scalar &loctime(), "\n";
print join("\t", &loctime()), "\n";

print "\nloctime(TS=>TIA); returns ", scalar &loctime(TS=>TIA), "\n";
print join("\t", &loctime(TS=>TIA)), "\n";

# Enhanced version of localtime(). Returns string time or list of time
# components just like localtime(). Accepts named options:
#
# TIME => time-in-secs  The time in seconds to convert, based on UTC time.
#                       The default is $^T.
# TS => time-standard  Adjust time for a different time standard. Time
#                       standards are UTC (the default), TIA, GPS-CORE.
#
# This routine returns undef if TZ or TS do not have acceptable values.
#
# NOTE: This routine does NOT accept a time-in-seconds like localtime()!
#       Use &loctime(TIME=>time()) or some such.

sub loctime {
    my (%params) = @_;          # Eat all the named parameters

    my ($t) = defined($params{TIME}) ? $params{TIME} : $^T;

    # Handle an optional time standard

    my ($ts) = $params{TS};
    if ($ts) {
        if ($ts eq "UTC") {      # Default, no time change needed
        } elsif ($ts eq "TIA") { $t -= 23;
        } elsif ($ts eq "GPS-CORE") { $t -= 6;
        } else {                 return undef;
        }
    }

    # TBD: Accept the TZ=>time-zone option.
    # The code below was a FAILED attempt to accept the TZ=>time-zone
    # option. It would return time for a different time-zone (EDT, PST, ...).
    #
    # local (%ENV) = %ENV;
    # if ($params{TZ}) { $ENV{"TZ"} = $params{TZ}; }
    #
    # It should work nicely, passing a changed environment to the perl
    # implementation of localtime(), EXCEPT that localtime is implemented
    # by the C runtime library, which (I guess) accesses the
    # environment directly. Drats.
}
```

```

# Return list or string based on the context of this call.
return wantarray ? (localtime ($t)) : localtime($t);
}

```

Output of hashcall.pl under MKS Toolkit 6.1 under Win98 using ActivePerl

```

sec   min   hour  mday  mon   year  wday  yday  isdst
loctime(); returns Tue Jun 20 08:29:20 2000
20    29    8     20    5     100   2     171   1
loctime(TS=>TIA); returns Tue Jun 20 08:28:57 2000
57    28    8     20    5     100   2     171   1

```

Sample of lines from Apache's error_log file

Each *group* of lines shown here is a *single* line in the error_log file. Also, there are no blank lines in the error_log file.

```
[Mon Jun  5 15:54:33 2000] [error] [client 216.52.234.14] File does not exist:
/usr/local/apache/htdocs/_themes/indust/indtextb.jpg
```

```
[Mon Jun  5 15:54:34 2000] [error] [client 216.52.234.14] File does not exist:
/usr/local/apache/htdocs/_derived/back_cmp_indust010_back_a.gif
```

```
[Mon Jun  5 17:53:08 2000] [crit] [client 216.107.135.182] (13)Permission denied:
/home/m/mm64/public_html/x52.9265/src/cxx_repository/.htaccess pcfg_openfile: unable to
check htaccess file, ensure it is readable
```

```
[Mon Jun  5 18:17:08 2000] [error] [client 199.229.1.2] File does not exist:
/home/c/cs89/public_html/HTMLOnLine/Frames/Frames.
```

Reading for Next Class

Llama book, Chapters 12, 13, 15, and 17

Homework Assignment

1. Write a program called webError.pl which implements this functionality:

```

sub usage {
    print <<END_OF_HELP;
usage: $0 [options]

```

Produce a summary report of web site errors from an Apache error log file.

Options:

```

-d          Turn on moderate debugging
-D          Turn on very verbose debugging
-f FILE     Specify the error file. Default is error_log
-h          Print this help message
-v          Turn on verbose reporting mode
END_OF_HELP
}

```

The file `error_log` is available at http://www.goss.com/perl/error_log. Your program should produce output that resembles this:

```
Web site errors from Mon Jun 5 15:54:33 2000 to Thu Jun 8 16:44:34 2000
```

```
Web server error file: error_log
Errors reported: 1000
```

```
Types of errors reported [followed by count]:
crit [57]
error [943]
```

In addition, with the `-v` option, your program should produce a nicely formatted report of more information from the `error_log` file. At the minimum, it should show each of the critical errors (ones with `[crit]` in the error). However, rather than showing duplicates of critical errors, it should show the error followed by a count in [brackets]. For example:

Critical errors, ordered by decreasing frequency:

```
(13)Permission denied: /home/s/ss427/public_html/.htaccess pcfg_openfile: unable to check
htaccess file, ensure it is readable [8]
```

```
(13)Permission denied: /home/a/aa547/public_html/assignment1/.htaccess pcfg_openfile:
unable to check htaccess file, ensure it is readable [8]
```

For extra credit, increase the usefulness of your `webError.pl` script. You can add more options, or improve the error reporting.

Hand in a printout of your modified program along *with sample output* at the beginning of next session.

3. Prepare a one-sentence answer to the question: “What are debuggers useful for?”¹

4. Download the handout for Session 6. This should be available at <http://www.goss.com/perl> by Wednesday, June 28 at 9AM. Bring a printout to Session 6.¹

Notes: ¹Not to be handed in.

This handout is Copyright © 2000 Clint Goss, All Rights Reserved.