

Korn Shell 93 Date Manipulation

Finnbarr P. Murphy

(fpm@fpmurphy.com)

While *bash* is the default shell on most, if not all, Linux distributions, there are times when using *ksh93* is more efficient and thus makes more sense. A classic problem in shell scripting is the manipulation of dates and times. Most shells do not include support for date/time string manipulation and the user is left to roll their own routines as needed. Typically this involves parsing date/time strings and using lookup tables and/or using a version of *date* with support for formatting date/time strings other than current date.

Since 1999, when version *h* of *ksh93* (the 1993 version of the Korn Shell) was released, *ksh93* has included such support via the *printf* builtin function. However examples on using this feature are scarce and I have written this short article in an attempt to make more shell script developers aware of this extremely useful and powerful feature in *ksh93*.

The *ksh93* builtin *printf* (not *printf(1)*) includes a *%T* formatting option.

```
%T          Treat argument as a date/time string and format it accordingly.
%(dateformat)T  T can be preceded by dateformat, where dateformat is any date format supported
                by the date(1) command.
```

Some examples will illustrate the power of this feature.

Output the current date just like the *date(1)* command.

```
$ printf "%T\n" now
Sat Mar 22 10:01:35 EST 2008
```

Output the current hour, minute and second.

```
$ printf "%(%H:%M:%S)T\n" now
10:02:07
```

Note that *ksh93* does not fork/exec the *date(1)* command to process this statement. It is built into *ksh93*. This results in faster shell script execution and less load on the operating system.

Output the number of seconds since the UNIX Epoch.

```
$ printf "%(%s)T\n" now
1206199251
```

If you know the number of seconds since the UNIX Epoch you can output the corresponding date/time in *ctime* format.

```
$ printf "%T\n" #1206199251
Sat Mar 22 10:22:35 EST 2008
```

The *printf* builtin also understands date/time strings like “2:00pm yesterday”, “this Wednesday”, “23 days ago”, “next 9:30am”, “in 6 days”, “+ 5 hours 10 minutes” and lots more. Look at the source code for the *printf* builtin (*.../cmd/ksh93/btins/print.c*) in the *ksh93* sources for more information on the various date/time strings which are supported.

Output the date/time corresponding to “2:00pm yesterday.”

```
$ printf "T\n" "2:00pm yesterday"
Fri Mar 21 14:00:00 EST 2008
```

Output the day of the week corresponding to the last day of February 2008.

```
$ printf "%(a)T\n" "final day Feb 2008"
Fri
```

Output the date corresponding to the third Wednesday in May 2008.

```
$ printf "%(D)T\n" "3rd wednesday may 2008"
05/21/08
```

Output what date it was 4 weeks ago.

```
$ printf "%(D)T\n" "4 weeks ago"
02/18/08
```

You can assign the output of *printf* “%T” to a variable. Note that “1997-198” represents the 198th day in 1997.

```
$ datestr=$(printf "%(D)T" "1997-198")
$ print $datestr
07/17/97
```

The *printf* builtin even understands *crontab* and *at* date/time syntax as the following two examples demonstrate.

Output the date/time the command associated with this *crontab* entry will next execute.

```
$ printf "%T\n" "0 0 1,15 * 1"
Mon Sep 1 00:00:00 EDT 2008
```

Output the date/time the command associated with this *at* date/time string will execute.

```
$ printf "%T\n" "exactly next hour"
Sun Mar 23 14:07:31 EST 2008
```

The following example shows how to output the date for the first and last days of last month. Care needs to be taken in the order in which the date string is entered as not all combinations are valid.

```
$ printf "%(%Y-%m-%d)T\n" "1st last month"
2008-05-01
$ printf "%(%Y-%m-%d)T\n" "final last month"
2008-05-31
```

Microseconds are also understood by `%T` formatting option. Note `%N` outputs 9 digits by default unless you limit output using a length specifier as in the following example.

```
$ datestr="2008-11-24 05:17:00.7043"
$ printf "%(%m-%d-%Y %T.%4N)T\n" "$datestr"
11-24-2008 05:17:00.7043
```

The next example is a short shell script which tackles a common problem associated with backing up files and deleting logs, i.e. calculate the difference between two given dates.

```
#
# USAGE: diffdate start-date finish-date
#
# EXAMPLE: diffdate "Tue, Feb 19, 2008 08:00:02 PM" "Wed, Feb 20, 2008 02:19:09 AM"
#
# Note: Maximum of 100 hours difference
#
SDATE=$(printf '%(%s)T' "$1")
FDATE=$(printf '%(%s)T' "$2")
[[ $# -ne 2 ]] {
    Usage: diffdate start-date finish-date&quot;
    exit 1
}
DIFF=$((FDATE-$SDATE))
SECS=$((DIFF % 60))
MINS=$((DIFF / 60 % 60))
HOURS=$((DIFF / (60 * 60)))
printf "%02d:%02d:%02d\n" $HOURS $MINS $SECS
```

My final example shows how to output a range of dates in a specific format incremented by 1 hour each time.

```
startdate="2008-05-26 01:00:00"
count=71
for ((i=0; i &lt; count; i++))
do
    printf "%(%m%d%Y%H0000)T\n" "${startdate} + $i hour"
done
```

Well, that is about all there is to the `printf %T` feature in `ksh93`. I hope that you have found this short article on date/time manipulation using this feature to be useful and informative and that you will start using it in your future Korn Shell scripts.

For personal use only