Command Line: date & cal

Time for a date

The command line tools cal and date are used to create calendars and edit dates and times. Used smartly, you can even create your own birthday reminder utility. BY HEIKE JURZIK

It does not always have to be a GUI tool or a PDA interface that helps you keep track of your appointments. Linux' *cal* and *date* functions can help you create your very own command line based calendar software. These tools are still the undisputed champions when it comes to shell scripts. After taking a short look at the various *cal* and *date* options, we will be moving on to a simple example that explains how a few easy steps can create a script that reminds you of your loved ones' birthdays when you log on to Linux.

What Year?

This simple program displays a basic calendar in the command line:

```
huhn@asteroid:~$ cal
     April 2003
Su Mo Tu We Th Fr Sa
       1  2  3  4  5
 6  7  8  9 10 11 12
13 14 15 16 17 18 19
20 21 22 23 24 25 26
27 28 29 30
```

Executing the command without any flags will simply display the calendar for the current month. However, the *-y* option outputs a calendar for the current year. Pipe the output through *less* (*cal -y | less*), to prevent the output from scrolling off screen. *cal* will display a specific month, if you additionally supply the month and year (four digits), as in *cal 06 2003* for June 2003. Don't make the mistake of supplying a two-digit year format; entering *cal 06 99* would not display June 1999, but June in the year 99 AD – *cal* goes back to January 1 0001.

The *-j* option tells *cal* to output the number of days since the beginning of the year (**Julian** representation):

```
huhn@asteroid:~$ cal -j
        April 2003
  S   M  Tu   W  Th   F   S
         91  92  93  94  95
[...]
```

The Right Time

The *date* command tells you the current date and time (for your timezone). The superuser *root* is additionally permitted to set the system time. Without any parameters *date* simply outputs the current system date and time:

```
huhn@asteroid:~$ date
Wed Apr 23 12:55:37 CEST 2003
```

If you want to change the English expressions for weeks and months, you can set the *LC_TIME* environment variable to a valid **Locale**. *export LC_TIME=de_DE* tells the *date* command to speak German, for example:

```
huhn@asteroid:~$ export LC_➥
TIME=de_DE
huhn@asteroid:~$ date
Mit Apr 23 13:55:32 CEST 2003
```

Environment variables can also be used to tell *date* to produce different output. The directory */usr/share/zoneinfo* contains an overview of the timezones, some of which are in subdirectories. You can temporarily change the *TZ* (for "TimeZone") variable to discover the time on say Easter Island:

```
huhn@asteroid:~$ TZ=Chile/➥
EasterIsland date
Mit Apr 23 11:34:26 EAST 2003
```

You can also assign different output formats to the date. To do so, use a plus sign, "+", followed by a string consisting of percent signs, "%", and letters indicating various format options. Table 1 provides an overview of the most important output formats.

You can specify *-d DATE* (or *--date=DATE*) to use the supplied *DATE*. Values such as *yesterday*, *next*, *3 days ago* or the full date (*09 June 1973*) are possible:

```
huhn@asteroid:~$ date -d ➥
"09 June 1973"
Sat Jun  9 00:00:00 CET 1973
```

Up to date!

The superuser *root* can use the *date* command to set the system date and time. The command expects a date/time value in *MMDDhhmmCCYY.ss* format, without a preceding plus character +, such as:

```
asteroid:~# date 04111723
Fri Apr 11 17:23:00 CEST 2003
```

The following values can be set:

- MM: Month, e.g. 04
- DD: Day, e.g. 13
- hh: Hour, e.g. 12
- mm: Minute, e.g. 59
- CC: the first two digits of the year (optional), e.g. 19
- YY: the second two digits of the year (optional), e.g. 73
- ss: seconds (optional), e.g. 24

An additional *-s* parameter allows a slightly less strict notation when specifying the time e.g. *date -s 04/29/2003*:

```
asteroid:~# date -s 04/29/2003
Tue Apr 29 00:00:00 CEST 2003
asteroid:~# date -s "04/29/➥
2003 15:58"
Tue Apr 29 15:58:00 CEST 2003
```

Note that changing the system time does not affect the **CMOS** clock. To adjust the CMOS clock to the current system time, you again need to log in as the superuser *root* and call *hwclock -w* (or the longer version *hwclock --systohc*, the option means "system to hardware clock").

Table 1: Output Formats for *date*

%%	The per cent character itself
%n	End of line
%t	Tab
%a	Weekday (short form), Output: *Mon*
%A	Weekday (long form), Output: *Monday*
%b	Name of month (short form), Output: *Apr*
%B	Name of month (long form), Output: *April*
%d	Day (two digits), Output: *01*
%e	Day (blank padded), Output: *1*
%D	Date in format *mm/dd/yy*, Output: *04/29/03*
%H	Hours (0 - 23)
%I	Hours (1 - 12)
%M	Minutes (00 - 59)
%p	am/pm, (AM, PM)
%r	Time, twelve hour clock (hh:mm:ss AM/PM)
%T	Time, 24 hour clock (hh:mm:ss)
%Z	Timezone

Listing 1: The Birthday Script

```
#!/bin/sh

today_cal=`date +%e`
echo -e "`cal | sed s/"${today_➥
cal}\>"/\\\\\\\\e[31\;7m\&\\\\\\\\e[39\;0m/g`"
echo
today=`date +%d.%m`
echo Today's date $today, birthday\(s\):
echo -e "\033[31m"; grep "^$today" ➥
~/.birthdays; echo -e "\033[0m"
nextday=`date -d tomorrow +%d.%m`
echo Tomorrow is $nextday, birthday\(s\):
grep "^$nextday" ~/.birthdays
echo
this_month=`date +%m`
echo In `date +%B` the following people ➥
will be celebrating their birthdays:
egrep "^[0-9]*\.$this_month\." ~/.birthdays
```

Happy Birthday to you!

The *date* command is really useful in shell scripts. The birthday diary (see Listing 1) shows how to use a neat combination of *cal* and *date*, based on a hidden file called *~/.birthdays* in your own home directory. The file has a date and a name in each line:

```
huhn@asteroid:~$ cat .birthdays
28.02.  Petronella
28.04.  Jones
27.04.  Easter
...
```

The *birthday.sh* script first assigns the output from *date +%e* to the *today_cal* variable (see also Table 1). The *echo -e* command that follows, enables the use of escape sequences, including the newline character, *\n*, and the tab *\t*. The *cal* command outputs the calendar and this output is piped ("|") to *sed*. *sed* then searches the calendar for the section containing the day specified by the output from *date +%e* and a word end (*>* is the regex for this). This is replaced by an *ESC[31;7m* (an ANSI escape sequence that switches the terminal to inverted red output), the selection itself (*&* is the *sed* reference to the replaced string) and *ESC[39;0m* (a second sequence that restores the standard terminal output mode). The numerous backslashes are required to mask these expressions for the shell and *sed*.

Following this, the output from *date +%d.%m* is assigned to the *today* variable. The script first outputs today's date, then switches the terminal to red (ANSI sequence). *grep* searches the *~/.birthdays* file for lines starting with *"^$today"*. Finally, the normal terminal output mode is restored.

The last line of the script can be interpreted as follows: the *egrep* program (a *grep* variant that can handle regular expressions) searches *~/.birthdays* for lines that start with any number of numbers (0 - 9), followed by a period, the current month and another period. The "^" character means "at the start of a line" and "[0-9]*" indicates any number of characters between 0 and 9. An entry in the following format

```
. ~/birthday.sh
```

in the *.profile* will run the script each time you log on – and that should help you keep an eye on those important dates.

THE AUTHOR

Heike Jurzik studied German, Computer Science and English at the University of Cologne, Germany. She discovered Linux in 1996 and has been fascinated with the scope of the Linux command line ever since. In her leisure time you might find Heike hanging out at Irish folk sessions or visiting Ireland.

GLOSSARY

Julian / Gregorian calendar: *The Julian calendar, a major reform of the Roman calendar, was introduced by Julius Caesar in the year 46 BC, adding an extra day every four years. These became known as leap years. The beginning of the year was moved to January 1. As this calendar did not precisely reflect astronomical time, Pope Gregory XIII introduced a reformed calendar in the year 1582 (ten days were dropped to compensate the calendar, and a new leap year format that dropped the extra day in leap years divisible by 100 by not by 400 was introduced). The Gregorian calendar was originally only introduced in Catholic countries, but gradually the new calendar did assert itself.*

Locales: *Linux supports a variety of output languages for program and system messages, dates or currency values. The locale command, "locale -a", will tell you the names of the locales installed on your system.*

CMOS: *Complimentary Metal-Oxide Semiconductor. Nearly all PCs have a battery driven clock that increments the date and time when the computer is powered off. As the clock is embedded in the same chip that has the CMOS RAM, it is referred to as a CMOS clock, or by one of the following common names: hardware clock, RTC (Real Time Clock) or BIOS clock.*