C2X Liaison: Removal of Deprecated Functions

Title: Removal of Deprecated Functions
Author: Nick Stoughton / The Austin Group
Date: 2020-08-17

As stated in N2528, The Austin Group is currently in the process of revising the POSIX (ISO/IEC Std 9945), and is trying to keep aligned with C17 and C2X changes as necessary.

There are several functions that are marked as "Obsolescent" in the current version of the POSIX standard that we would typically remove during this revision process. However, we note that in the current working draft for C2X the two obsolete functions "asctime_r()" and "ctime_r()" have been added.

During the August 2020 WG 14 meeting, there was general agreement to seek a paper that more closely aligned the C2X functions with POSIX, noting that it was at very least odd that C2X should be adding functions already marked as obsolete.

However, there was also concern that strftime, the proposed alternative to use for these new “asctime_r()” and “ctime_r()” , would “pull in locale baggage not wanted in small embedded programs”.

Locale Free Alternatives

It should be noted that asctime_r() is currently defined to be equivalent to a simple snprintf() call, and anyone who really wants a locale free way to print the date and time could use this.

As an experiment, three equivalent simple programs were constructed:

date1.c

```
#include <stdio.h>
#include <time.h>

char *
date1(const struct tm * timeptr , char * buf)
{
        static const char wday_name[7][3] = {
                "Sun", "Mon", "Tue", "Wed", "Thu", "Fri", "Sat"
        };
```

```
        static const char mon_name[12][3] = {
                "Jan", "Feb", "Mar", "Apr", "May", "Jun",
                "Jul", "Aug", "Sep", "Oct", "Nov", "Dec"
        };
        snprintf ( buf , 26, "%.3s %.3s%3d %.2d:%.2d:%2d %d\n",
                wday_name[timeptr->tm_wday],
                mon_name[timeptr->tm_mon],
                timeptr->tm_mday, timeptr->tm_hour,
                timeptr->tm_min, timeptr->tm_sec,
                1900 + timeptr->tm_year);
        return buf ;
}

int
main(int argc, const char *argv[])
{
        char buf[26];
        time_t t = time(NULL);

        printf("%s", date1(localtime(&t), buf));
        return 0;
}
```

date2.c

```
#include <stdio.h>
#include <time.h>

int
main(int argc, const char *argv[])
{
        char buf[26];
        time_t t = time(NULL);

        printf("%s", asctime_r(localtime(&t), buf));
        return 0;
}
```

date3.c

```
#include <stdio.h>
#include <time.h>

int
main(int argc, const char *argv[])
{
        char buf[26];
        time_t t = time(NULL);
        struct tm *tm = localtime(&t);

        strftime(buf, sizeof(buf), "%a %b %e %T %Y\n", tm);
        printf("%s", buf);
        return 0;
}
```

These three examples were then compiled with gcc 9.3.0 at -O3 optimization on an Ubuntu Linux platform.

Using the “size” utility, we see:

```
   text        data         bss         dec         hex     filename
   2377         632           8        3017         bc9     date1
   2063         632           8        2703         a8f     date2
   2102         632           8        2742         ab6     date3
```

Interestingly, the “snprintf” variant was the largest (and generated some “Note” level messages that the string could overflow if the year was out of normal range). The strftime() version is only about 40 bytes (1.4%) larger than the asctime_r() version.

The original functions, asctime() and ctime(), were never marked as obsolete in C (though they have been marked as such in POSIX since Issue 7 in 2008), so it seems wrong to simply remove them both from the C standard. Instead they should be marked as obsolescent.

The new functions (asctime_r() and ctime_r()) do offer a safer API, with guarantees of data race safety etc.

Therefore this proposal offers two alternatives. The first version simply keeps what is there in the current draft, but makes it obsolete. Since it does seem unusual to ask implementations to provide these functions but simultaneously recommending that nobody uses them, a second version is offered that returns to the C17 wording, and marks those as obsolete.

Proposed Wording Changes, Version 1

Change “7.27.3.1 The asctime functions”:

Replace the Synopsis with:

Synopsis

```
#include <time.h>
```

```
[[deprecated]] char * asctime(const struct tm *timeptr);
[[deprecated]] char * asctime_r(const struct tm *timeptr, char *buf);
```

Before paragraph 2, insert a new paragraph:

> These functions are obsolescent and should be avoided in new code.

Similarly for “7.27.3.2 The ctime functions”, replace the Synopsis with:

Synopsis

```
#include <time.h>
[[deprecated]] char * ctime(const time_t *timer);
[[deprecated]] char * ctime_r(const time_t *timer, char *buf);
```

Add a new paragraph before paragraph 2:

> These functions are obsolescent and should be avoided in new code.

Proposed Wording Changes, Version 2

Change 7.27.3.1 in its entirety (note this is the wording from C17, plus the deprecated attribute and obsolescent warning):

> **7.27.3.1 The asctime function**
>
> **Synopsis**
>
> ```
> #include <time.h>
> [[deprecated]] char * asctime(const struct tm *timeptr);
> ```
>
> **Description**
>
> This function is obsolescent and should be avoided in new code.
>
> The **`asctime`** function converts the broken-down time in the structure pointed to by `timeptr` into a string in the form
>
> ```
> Sun Sep 16 01:03:52 1973\n\0
> ```

using the equivalent of the following algorithm.

```
[[deprecated]] char * asctime(const struct tm *timeptr)
{
    static const char wday_name[7][3] = {
        "Sun", "Mon", "Tue", "Wed", "Thu", "Fri", "Sat"
    };
    static const char mon_name[12][3] = {
        "Jan", "Feb", "Mar", "Apr", "May", "Jun",
        "Jul", "Aug", "Sep", "Oct", "Nov", "Dec"
    };
    static char result[26];

    sprintf(result, "%.3s %.3s%3d %.2d:%.2d:%.2d %d\n",
        wday_name[timeptr->tm_wday],
        mon_name[timeptr->tm_mon],
        timeptr->tm_mday, timeptr->tm_hour,
        timeptr->tm_min, timeptr->tm_sec,
        1900 + timeptr->tm_year);
    return result;
}
```

If any of the members of the broken-down time contain values that are outside their normal ranges*, the behavior of the **`asctime`** function is undefined. Likewise, if the calculated year exceeds four digits or is less than the year 1000, the behavior is undefined.

Returns

The asctime function returns a pointer to the string.

*) See 7.27.1

Change 7.27.3.2 in its entirety (note this is the wording from C17, plus the deprecated attribute and obsolescent warning):

7.27.3.2 The `ctime` function

Synopsis

```
#include <time.h>
[[deprecated]] char * ctime(const time_t *timer);
```

Description

This function is obsolescent and should be avoided in new code.

The ctime function converts the calendar time pointed to by timer to local time in the form of a string. It is equivalent to

```
asctime(localtime(timer))
```

Returns

The **ctime** function returns the pointer returned by the **asctime** function with that broken-down time as argument.

Acknowledgements

I would like to thank the core Austin Group for reviewing this, and especially to Geoff Clare.