```
 1: // $Id: futuretimes.c,v 1.44 2014-02-20 15:28:20-08 - - $
 2:
 3: #include <limits.h>
 4: #include <locale.h>
 5: #include <stdio.h>
 6: #include <stdlib.h>
 7: #include <time.h>
 8:
 9: #define SEC_PER_DAY (24 * 60 * 60)
10: #define SEC_PER_YEAR (SEC_PER_DAY * 365 + SEC_PER_DAY / 4)
11: const time_t localtime_segfault = (time_t) INT_MAX * SEC_PER_YEAR;
12:
13: int main (void) {
14:    setlocale (LC_NUMERIC, "en_US");
15:    for (time_t when = 1; when > 0; when = when * 4 + 3) {
16:       if (when > localtime_segfault) break;
17:       char *timeformat = "%b %d %a %H:%M:%S %Z";
18:       char buffer[256];
19:       struct tm *tm = localtime (&when);
20:       strftime (buffer, sizeof buffer, timeformat, tm);
21:       printf ("%'25ld = %'15d %s\n", when, tm->tm_year + 1900, buffer);
22:    }
23:    printf ("%'25ld = %'15d = INT_MAX * SEC_PER_YEAR\n",
24:             localtime_segfault, INT_MAX);
25:    printf ("%'25ld = %'15ld = LONG_MAX\n", LONG_MAX,
26:             LONG_MAX / SEC_PER_YEAR);
27:    return EXIT_SUCCESS;
28: }
29:
30: //TEST// ./futuretimes >futuretimes.lis
31: //TEST// mkpspdf futuretimes.ps futuretimes.c* futuretimes.lis
32:
```

```
1: @@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@ mkc: starting futuretimes.c
2: futuretimes.c:
3:      $Id: futuretimes.c,v 1.44 2014-02-20 15:28:20-08 - - $
4: gcc -g -O0 -Wall -Wextra -rdynamic -std=gnu11 futuretimes.c
5:         -o futuretimes -lglut -lGLU -lGL -lX11 -lrt -lm
6: rm -f futuretimes.o
7: @@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@ mkc: finished futuretimes.c
```

```
 1:                             1 =           1,969 Dec 31 Wed 16:00:01 PST
 2:                             7 =           1,969 Dec 31 Wed 16:00:07 PST
 3:                            31 =           1,969 Dec 31 Wed 16:00:31 PST
 4:                           127 =           1,969 Dec 31 Wed 16:02:07 PST
 5:                           511 =           1,969 Dec 31 Wed 16:08:31 PST
 6:                         2,047 =           1,969 Dec 31 Wed 16:34:07 PST
 7:                         8,191 =           1,969 Dec 31 Wed 18:16:31 PST
 8:                        32,767 =           1,970 Jan 01 Thu 01:06:07 PST
 9:                       131,071 =           1,970 Jan 02 Fri 04:24:31 PST
10:                       524,287 =           1,970 Jan 06 Tue 17:38:07 PST
11:                     2,097,151 =           1,970 Jan 24 Sat 22:32:31 PST
12:                     8,388,607 =           1,970 Apr 07 Tue 18:10:07 PST
13:                    33,554,431 =           1,971 Jan 24 Sun 00:40:31 PST
14:                   134,217,727 =           1,974 Apr 03 Wed 03:42:07 PDT
15:                   536,870,911 =           1,987 Jan 05 Mon 10:48:31 PST
16:                 2,147,483,647 =           2,038 Jan 18 Mon 19:14:07 PST
17:                 8,589,934,591 =           2,242 Mar 16 Wed 05:56:31 PDT
18:                34,359,738,367 =           3,058 Oct 25 Mon 20:46:07 PDT
19:               137,438,953,471 =           6,325 Apr 08 Wed 08:04:31 PDT
20:               549,755,813,887 =          19,391 Jan 25 Tue 04:18:07 PST
21:             2,199,023,255,551 =          71,654 Apr 09 Thu 18:12:31 PDT
22:             8,796,093,022,207 =         280,707 Feb 03 Sun 20:50:07 PST
23:            35,184,372,088,831 =       1,116,918 May 14 Sat 12:20:31 PDT
24:           140,737,488,355,327 =       4,461,763 Jun 19 Sun 22:22:07 PDT
25:           562,949,953,421,311 =      17,841,143 Nov 13 Sat 13:28:31 PST
26:         2,251,799,813,685,247 =      71,358,665 Jun 19 Mon 05:54:07 PST
27:         9,007,199,254,740,991 =     285,428,751 Nov 11 Sun 23:36:31 PST
28:        36,028,797,018,963,967 =   1,141,709,097 Jun 12 Sat 22:26:07 PST
29:        67,769,429,938,567,200 =   2,147,483,647 = INT_MAX * SEC_PER_YEAR
30: 9,223,372,036,854,775,807 =     292,271,023,045 = LONG_MAX
```