

(Downloaded from <https://cs.stanford.edu/~knuth/programs.html> and typeset on May 28, 2023)

1. Extended Hello World program. This is a medium-short demonstration of CWEB.

```

⟨ Include files 3 ⟩
⟨ Global variables 9 ⟩
⟨ Subroutines 8 ⟩
main()
{
    ⟨ Local variables 5 ⟩;
    ⟨ Print a greeting 2 ⟩;
    ⟨ Print the date and time 4 ⟩;
    ⟨ Print an interesting date and time from the past 6 ⟩;
}

```

2. First we brush up our Shakespeare by quoting from *The Merchant of Venice* (Act I, Scene I, Line 77). This makes the program literate.

```

⟨ Print a greeting 2 ⟩ ≡
    printf("Greetings...to\n");    /* Hello, */
    printf(" 'the stage,");        /* world */
    printf("where every man must play a part'.\n");

```

This code is used in section 1.

3. Since we're using the *printf* routine, we had better include the standard input/output header file.

```

⟨ Include files 3 ⟩ ≡
#include <stdio.h>

```

See also section 7.

This code is used in section 1.

4. Now we brush up our knowledge of C runtime routines, by recalling that the function *time*(0) returns the current time in seconds.

```

⟨ Print the date and time 4 ⟩ ≡
    current_time = time(0);
    printf("Today is");
    print_date(current_time);
    printf(",\n and the time is");
    print_time(current_time);
    printf("\n");

```

This code is used in section 1.

5. The value returned by *time*(0) is, more precisely, a value of type **time_t**, representing seconds elapsed since 00:00:00 Greenwich Mean Time on January 1, 1970.

At present, **time_t** is equivalent to **long**. But a 32-bit integer will become too small to hold the result of *time*(0) on January 18, 2038, at 19:14:08, Pacific Standard Time. We will try to write a program that will still work on January 19, 2038 (although it will have to be recompiled), by declaring *current_time* to have type **time_t** instead of type **long**.

```

⟨ Local variables 5 ⟩ ≡
    time_t current_time;    /* seconds after the epoch */

```

This code is used in section 1.

6. Ten million minutes is 600,000,000 seconds.

⟨ Print an interesting date and time from the past 6 ⟩ ≡
printf("Ten_million_minutes_ago_it_was\n");
print_date(*current_time* - 600000000);
printf(",at");
print_time(*current_time* - 600000000);
printf".\n");

This code is used in section 1.

7. Date and time. The remaining task is to write subroutines to print dates and times.

UNIX's *localtime* function does most of the work for us, but we need to include another system header file before we can use it.

```
<Include files 3> +=
#include <time.h>
```

8. First, let's work on the date. We want to produce an American-style date such as "Monday, January 18, 2038".

The result of *localtime* is a pointer to a *tm* structure, which has 11 fields, as explained in the man page for *ctime*(3V). For example, one of the fields is *tm_year*, the year minus 1900.

Note that the parameter to *localtime* must be a pointer to the time in seconds, not the time itself.

```
<Subroutines 8> ≡
print_date(clk)
    time_t clk;    /* seconds since the epoch */
{
    struct tm *t;    /* data deduced from clk */
    t = localtime(&clk);
    printf("%s, %s %d, %d",
        day_name[t->tm_wday], month_name[t->tm_mon],
        t->tm_mday, t->tm_year + 1900);
}
```

See also section 10.

This code is used in section 1.

```
9. <Global variables 9> ≡
char *day_name[] = {"Sunday", "Monday", "Tuesday", "Wednesday", "Thursday", "Friday",
    "Saturday"};
char *month_name[] = {"January", "February", "March", "April", "May", "June", "July", "August",
    "September", "October", "November", "December"};
```

This code is used in section 1.

10. The subroutine for time is similar to the routine for date. We could make use of the fact that *print_time* is always called after *print_date*, with the same parameter; that would save a call on *localtime*. But let's make the subroutine more general, so that we can use it later in another program.

```
<Subroutines 8> +=
print_time(clk)
    time_t clk;    /* seconds since the epoch */
{
    struct tm *t;    /* data deduced from clk */
    t = localtime(&clk);
    <Print the hours and minutes 11>;
    <Print "am" or "pm" as appropriate 12>;
    printf(", %s", t->tm_zone);
}
```

11. The tricky thing here is to make 0 hours come out as 12, yet 13 is changed to 1. If the number of minutes is less than 10, we want a leading zero to be printed.

```
<Print the hours and minutes 11> ≡
printf("%d:%02d", ((t->tm_hour + 11) % 12) + 1, t->tm_min);
```

This code is used in section 10.

12. Instead of trying to figure out whether noon and midnight are “am” or “pm,” we treat them as special cases.

⟨ Print “am” or “pm” as appropriate 12 ⟩ \equiv

```
    if ( $t\text{-}tm\_min \equiv 0 \wedge (t\text{-}tm\_hour \% 12) \equiv 0$ ) printf("%s",  $t\text{-}tm\_hour \equiv 0 ? \text{"midnight"} : \text{"noon"}$ );  
    else printf("%s",  $t\text{-}tm\_hour < 12 ? \text{"am"} : \text{"pm"}$ );
```

This code is used in section 10.

13. Index. CWEB prepares an index that shows where each identifier is used and/or declared.

clk: [8](#), [10](#).
ctime: [8](#).
current_time: [4](#), [5](#), [6](#).
day_name: [8](#), [9](#).
localtime: [7](#), [8](#), [10](#).
main: [1](#).
month_name: [8](#), [9](#).
print_date: [4](#), [6](#), [8](#), [10](#).
print_time: [4](#), [6](#), [10](#).
printf: [2](#), [3](#), [4](#), [6](#), [8](#), [10](#), [11](#), [12](#).
t: [8](#), [10](#).
time: [4](#), [5](#).
tm: [8](#), [10](#).
tm_hour: [11](#), [12](#).
tm_mday: [8](#).
tm_min: [11](#), [12](#).
tm_mon: [8](#).
tm_wday: [8](#).
tm_year: [8](#).
tm_zone: [10](#).

⟨ Global variables [9](#) ⟩ Used in section [1](#).
⟨ Include files [3](#), [7](#) ⟩ Used in section [1](#).
⟨ Local variables [5](#) ⟩ Used in section [1](#).
⟨ Print “am” or “pm” as appropriate [12](#) ⟩ Used in section [10](#).
⟨ Print a greeting [2](#) ⟩ Used in section [1](#).
⟨ Print an interesting date and time from the past [6](#) ⟩ Used in section [1](#).
⟨ Print the date and time [4](#) ⟩ Used in section [1](#).
⟨ Print the hours and minutes [11](#) ⟩ Used in section [10](#).
⟨ Subroutines [8](#), [10](#) ⟩ Used in section [1](#).

HWTIME

	Section	Page
Extended Hello World program	1	1
Date and time	7	3
Index	13	5