Next: Processor And CPU Time, Previous: Time Basics, Up: Date and Time [Contents][Index]

21.2 Elapsed Time

One way to represent an elapsed time is with a simple arithmetic data type, as with the following function to compute the elapsed time between two calendar times. This function is declared in `time.h`.

Function: *double* **difftime** *(time_t time1, time_t time0)*

> Preliminary: | MT-Safe | AS-Safe | AC-Safe | See POSIX Safety Concepts.
>
> The `difftime` function returns the number of seconds of elapsed time between calendar time *time1* and calendar time *time0*, as a value of type `double`. The difference ignores leap seconds unless leap second support is enabled.
>
> In the GNU C Library, you can simply subtract `time_t` values. But on other systems, the `time_t` data type might use some other encoding where subtraction doesn't work directly.

The GNU C Library provides two data types specifically for representing an elapsed time. They are used by various GNU C Library functions, and you can use them for your own purposes too. They're exactly the same except that one has a resolution in microseconds, and the other, newer one, is in nanoseconds.

Data Type: **struct timeval**

> The `struct timeval` structure represents an elapsed time. It is declared in `sys/time.h` and has the following members:
>
> `time_t tv_sec`
>
> > This represents the number of whole seconds of elapsed time.
>
> `long int tv_usec`
>
> > This is the rest of the elapsed time (a fraction of a second), represented as the number of microseconds. It is always less than one million.

Data Type: **struct timespec**

> The `struct timespec` structure represents an elapsed time. It is declared in `time.h` and has the following members:
>
> `time_t tv_sec`
>
> > This represents the number of whole seconds of elapsed time.
>
> `long int tv_nsec`
>
> > This is the rest of the elapsed time (a fraction of a second), represented as the number of nanoseconds. It is always less than one billion.

It is often necessary to subtract two values of type `struct timeval` or `struct timespec`. Here is the best

way to do this. It works even on some peculiar operating systems where the `tv_sec` member has an unsigned type.

```
/* Subtract the 'struct timeval' values X and Y,
   storing the result in RESULT.
   Return 1 if the difference is negative, otherwise 0. */

int
timeval_subtract (result, x, y)
     struct timeval *result, *x, *y;
{
  /* Perform the carry for the later subtraction by updating y. */
  if (x->tv_usec < y->tv_usec) {
    int nsec = (y->tv_usec - x->tv_usec) / 1000000 + 1;
    y->tv_usec -= 1000000 * nsec;
    y->tv_sec += nsec;
  }
  if (x->tv_usec - y->tv_usec > 1000000) {
    int nsec = (x->tv_usec - y->tv_usec) / 1000000;
    y->tv_usec += 1000000 * nsec;
    y->tv_sec -= nsec;
  }

  /* Compute the time remaining to wait.
     tv_usec is certainly positive. */
  result->tv_sec = x->tv_sec - y->tv_sec;
  result->tv_usec = x->tv_usec - y->tv_usec;

  /* Return 1 if result is negative. */
  return x->tv_sec < y->tv_sec;
}
```

Common functions that use `struct timeval` are `gettimeofday` and `settimeofday`.

There are no GNU C Library functions specifically oriented toward dealing with elapsed times, but the calendar time, processor time, and alarm and sleeping functions have a lot to do with them.

Next: Processor And CPU Time, Previous: Time Basics, Up: Date and Time [Contents][Index]