Subsections

Time Functions

In this chapter we will look at how we can access the clock time with UNIX system calls.

There are many more time functions than we consider here - see `man` pages and standard library function listings for full details. In this chapter we concentrate on applications of timing functions in C

Uses of time functions include:

- telling the time.
- timing programs and functions.
- setting number seeds.

Basic time functions

Some of thge basic time functions are prototypes as follows:

`time_t time(time_t *tloc)` -- returns the time since `00:00:00 GMT, Jan. 1, 1970`, measured in seconds.

If `tloc` is not NULL, the return value is also stored in the location to which tloc points.

`time()` returns the value of time on success.

On failure, it returns `(time_t) -1`. `time_t` is typedefed to a long (int) in <sys/types.h> and <sys/time.h> header files.

`int ftime(struct timeb *tp)` -- fills in a structure pointed to by `tp`, as defined in <sys/timeb.h>:

```
struct timeb
                { time_t time;
                                        unsigned short millitm;
                                        short timezone;
                                        short dstflag;
                                    };
```

The structure contains the time since the epoch in seconds, up to 1000 milliseconds of more precise interval, the local time zone (measured in minutes of time westward from Greenwich), and a flag that, if nonzero, indicates that Day light Saving time applies locally during the appropriate part of the year.

On success, `ftime() returns no useful value. On failure, it returns -1.`

`Two other functions defined` ***`etc.`*** `in #include <time.h>`

```
char *ctime(time_t *clock),
char *asctime(struct tm *tm)

ctime() converts a long integer, pointed to by clock, to a 26-character
string of the form produced by asctime(). It first breaks down clock to a
tm structure by calling localtime(), and then calls asctime() to convert
that tm structure to a string.

asctime() converts a time value contained in a tm structure to a
26-character string of the form:

   Sun Sep 16 01:03:52 1973

asctime() returns a pointer to the string.
```

Example time applications

we mentioned above three possible uses of time functions (there are many more) but these are very common.

Example 1: Time (in seconds) to perform some computation

This is a simple program that illustrates that calling the time function at distinct moments and noting the different times is a simple method of timing fragments of code:

```
/* timer.c */

#include <stdio.h>
#include <sys/types.h>
#include <time.h>

main()
  {  int i;
                                    time_t t1,t2;

                                    (void) time(&t1);
                                    for (i=1;i<=300;++i)
                                      printf(``%d %d %d\n'',i, i*i, i*i*i);

                                     (void) time(&t2);
                                     printf(``\n Time to do 300 squares and

                                    cubes= %d seconds\n'', (int) t2-t1);

                  }
```

Example 2: Set a random number seed

We have seen a similar example previously, this time we use the `lrand48()` function to generate of number sequence:

```
/* random.c */
#include <stdio.h>
#include <sys/types.h>
#include <time.h>
```

```
main()
  { int i;
                                    time_t t1;

                                    (void) time(&t1);
                                    srand48((long) t1);
                                    /* use time in seconds to set seed */
                                    printf(``5 random numbers
                                      (Seed = %d):\n'',(int) t1);

                                    for (i=0;i<5;++i)
                                      printf(``%d '', lrand48());
                                    printf(``\n\n''); /* flush print buffer */

                }
```

lrand48() returns non-negative long integers uniformly distributed over the interval (0, 2**31).

A similar function drand48() returns double precision numbers in the range [0.0,1.0).

srand48() sets the seed for these random number generators. It is important to have different seeds when we call the functions otherwise the same set of pseudo-random numbers will generated. time() always provides a unique seed.

Exercises

Exercise 12708

Write a C program that times a fragment of code in milliseconds.

Exercise 12709

Write a C program to produce a series of floating point random numbers in the ranges (a) 0.0 - 1.0 (b) 0.0 - n where n is any floating point value. The seed should be set so that a unique sequence is guaranteed.

Dave Marshall
1/5/1999