[successivo] [precedente] [inizio] [fine] [indice generale] [indice ridotto] [indice analitico] [home] [volume] [parte]

Capitolo 604. C: «time.h»

Il file `'time.h'` della libreria standard definisce principalmente delle funzioni per il trattamento delle informazioni data-orario. Non è stabilito in che modo venga rappresentato il tempo internamente alle funzioni, anche se di norma si tratta di un valore intero che esprime una quantità di secondi o di frazioni di secondo.

604.1 Il tempo di CPU

La funzione **`'clock()'`** consente di ottenere il tempo di utilizzo del microprocessore (CPU), espresso virtualmente in cicli di CPU. In pratica, viene definita la macro-variabile **`'CLOCKS_PER_SEC'`**, contenente il valore che esprime convenzionalmente la quantità di cicli di CPU per secondo; quindi, il valore restituito dalla funzione **`'clock()'`** si traduce in secondi dividendolo per **`'CLOCKS_PER_SEC'`**. Il valore restituito dalla funzione **`'clock()'`** e l'espressione in cui si traduce la macro-variabile **`'CLOCKS_PER_SEC'`** sono di tipo **`'clock_t'`**:

```
typedef long int clock_t;   // Unità di tempo convenzionale che
                            // rappresenta un ciclo virtuale di CPU.

#define CLOCKS_PER_SEC  1000000L  // Valore convenzionale di 1 s, in
                                  // termini di cicli virtuali di CPU.

clock_t clock (void);             // Tempo di utilizzo della CPU.
```

La funzione **`'clock()'`** restituisce il tempo di CPU espresso in unità **`'clock_t'`**, utilizzato dal processo elaborativo a partire dall'avvio del programma. Se la funzione non è in grado di dare questa indicazione, allora restituisce il valore -1, o più precisamente **`'(clock_t) (-1)'`**.

Per valutare l'intervallo di tempo di utilizzo della CPU, da una certa posizione del programma, a un'altra, occorre memorizzare i valori ottenuti dalla funzione e poi procedere a una sottrazione.

Per comprendere il significato della funzione **`'clock()'`**, del tipo **`'clock_t'`** e della macro-variabile **`'CLOCKS_PER_SEC'`**, viene proposto un esempio molto semplice, ma completo, dove si intende che il tipo **`'clock_t'`** sia intero e sia contenibile in una variabile di tipo **`'long int'`**:

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

int
main (int argc, char *argv[])
{
    clock_t t0;
    clock_t t1;
    long int i;
    long int x;

    t0 = clock ();
    printf ("Tempo iniziale: %ld/%ld\n",
            (long int) t0, (long int) CLOCKS_PER_SEC);

    for (i = 0; i < 10000000; i++)
      {
        x = i * 123;
      }

    t1 = clock ();
    printf ("Tempo finale: %ld/%ld\n",
            (long int) t1, (long int) CLOCKS_PER_SEC);

    return 0;
}
```

Avviando questo programma si potrebbe leggere un risultato simile al testo, dove si vede un valore di **'CLOCKS_PER_SEC'** pari a 1 000 000:

```
Tempo iniziale: 0/1000000
Tempo finale: 20000/1000000
```

604.2 Rappresentazione interna del tempo

Generalmente, nei sistemi Unix si tratta il tempo come una quantità di secondi trascorsi a partire da un'epoca di riferimento, che tradizionalmente coincide con l'ora zero del giorno 1 gennaio 1970. Da questo concetto deriva il tipo **'time_t'** della libreria, che, secondo lo standard, rappresenta la quantità di unità di tempo trascorsa a partire da un'epoca di riferimento.

```
typedef long int time_t;        // Unità di tempo convenzionale
                                // per le informazioni data-orario.
```

Ammesso che si tratti di un numero intero, così come viene ipotizzato

dall'esempio proposto, il rango costituisce il limite alle date rappresentabili. Pertanto, se il tipo `'time_t'` viene dichiarato come numero intero con segno, a 32 bit, per rappresentare una quantità di secondi (come nella tradizione Unix), significa che si possono rappresentare al massimo 24 855 giorni, pari a circa 68 anni.(1) Se l'epoca di riferimento è il 1970, si può arrivare al massimo al 2038.

604.3 Rappresentazione strutturata del tempo

La libreria standard prescrive che sia definito il tipo `'struct tm'`, con il quale è possibile rappresentare tutte le informazioni relative a un certo tempo, secondo le convenzioni umane. Lo standard prescrive con precisione i membri minimi della struttura e l'intervallo di valori che possono contenere:

```
struct tm {
    int tm_sec;     // Secondi:                da 0 a 60.
    int tm_min;     // Minuti:                 da 0 a 59.
    int tm_hour;    // Ora:                    da 0 a 23.
    int tm_mday;    // Giorno del mese:        da 1 a 31.
    int tm_mon;     // Mese dell'anno:         da 0 a 11.
    int tm_year;    // Anno dal 1900.
    int tm_wday;    // Giorno della settimana: da 0 a 6
                    // con lo zero corrispondente alla domenica.
    int tm_yday;    // Giorno dell'anno:       da 0 a 365.
    int tm_isdst;   // Ora estiva. Contiene un valore positivo
                    // se è in vigore l'ora estiva; zero se l'ora
                    // è quella «normale» ovvero quella invernale;
                    // un valore negativo se l'informazione non è
                    // disponibile.
};
```

Si può osservare che il mese viene rappresentato con valori che vanno da 0 a 11, pertanto gennaio si indica con lo zero e dicembre con il numero 11; inoltre, l'intervallo ammesso per i secondi consente di rappresentare un secondo in più, dato che l'intervallo corretto sarebbe da 0 a 59; infine, il fatto che i giorni dell'anno vadano da 0 (il primo) a 365 (l'ultimo), significa che negli anni normali i valori vanno da 0 a 364, mentre negli anni bisestili si arriva a contare fino a 365.

604.4 Funzioni per l'elaborazione di valori legati al tempo

Un gruppo di funzioni dichiarate nel file `'time.h'` ha lo scopo di elaborare in qualche modo le informazioni legate al tempo ed eventualmente di convertirle in formati diversi. Queste funzioni trattano il tempo in forma di variabili di tipo

'time_t' o di tipo **'struct tm'**.[(2)]

La variabile di tipo **'time_t'** che viene usata in queste funzioni potrebbe esprimere un valore riferito al tempo universale (UT), mentre le funzioni che la utilizzano dovrebbero tenere conto del fuso orario, in base alle informazioni che può offrire il sistema operativo.

604.4.1 Funzione «time()»

La funzione **'time()'** determina il tempo attuale secondo il calendario del sistema operativo, restituendolo nella forma del tipo **'time_t'**. La funzione richiede un parametro, costituito da un puntatore di tipo **'time_t *'**: se questo puntatore è valido, la stessa informazione che viene restituita viene anche memorizzata nell'indirizzo indicato da tale puntatore.

```
time_t  time (time_t *timer);
```

In pratica, se è possibile, l'informazione data-orario raccolta dalla funzione, viene anche memorizzata in **timer*.

Se la funzione non può fornire l'informazione richiesta, allora restituisce il valore -1, o più precisamente: **'(time_t) (-1)'**.

604.4.2 Funzione «difftime()»

La funzione **'difftime()'** calcola la differenza tra due date, espresse in forma **'time_t'** e restituisce l'intervallo in secondi, in una variabile in virgola mobile, di tipo **'double'**:

```
double  difftime (time_t time1, time_t time0);
```

Per la precisione, viene eseguito *time1-time0* e di conseguenza va il segno del risultato.

604.4.3 Funzione «mktime()»

La funzione **'mktime()'** riceve come argomento il puntatore a una variabile strutturata di tipo **'struct tm'**, contenente le informazioni sull'ora locale, e determina il valore di quella data secondo la rappresentazione interna, di tipo **'time_t'**:

```
time_t  mktime (struct tm *timeptr);
```

La funzione tiene in considerazione solo alcuni membri della struttura; per la precisione, non considera il giorno della settimana e il giorno dell'anno; inoltre, ammette anche valori al di fuori degli intervalli stabiliti per i vari membri della struttura; infine, considera un valore negativo per il membro '***timeptr*->tm_isdst**' come la richiesta di determinare se sia o meno in vigore l'ora estiva per la data indicata.

Se la funzione non è in grado di restituire un valore rappresentabile nel tipo '**time_t**', o comunque se non può eseguire il suo compito, restituisce il valore -1, o più precisamente '**(time_t) (-1)**'. Se invece tutto procede regolarmente, la funzione provvede anche a correggere i valori dei vari membri della struttura e a ricalcolare il giorno della settimana e dell'anno.

L'esempio successivo mostra la dichiarazione di una variabile strutturata di tipo '**struct tm**', assegnando ai suoi membri dei valori non corretti. Con l'aiuto della funzione '**mktime()**' si ricostruisce la data secondo le convenzioni comuni:

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

int
main (int argc, char *argv[])
{
    struct tm t;
    time_t tx;

    t.tm_year = 107;    // 2007 - 1900
    t.tm_mon  = 5;
    t.tm_mday = 33;
    t.tm_hour = 0;
    t.tm_min  = 0;
    t.tm_sec  = 60;
    t.tm_isdst = -1;

    printf ("%d/%d/%d %d:%d:%d\n",
            t.tm_year + 1900, t.tm_mon + 1, t.tm_mday,
            t.tm_hour, t.tm_min, t.tm_sec);

    tx = mktime (&t);

    if (tx == (time_t) (-1))
      {
        printf ("Errore! %ld\n", (long int) tx);
      }
    else
```

```
      {
        printf ("%d/%d/%d %d:%d:%d\n",
                 t.tm_year + 1900, t.tm_mon + 1, t.tm_mday,
                 t.tm_hour, t.tm_min, t.tm_sec);

        printf ("giorno della settimana: %d\n", t.tm_wday);
        printf ("giorno dell'anno: %d\n", t.tm_yday + 1);
        printf ("ora estiva: %d\n", t.tm_isdst);
      }
    return 0;
}
```

Eseguendo questo programma di esempio si dovrebbe ottenere il testo seguente:

```
2007/6/33 0:0:60
2007/7/3 0:1:0
giorno della settimana: 2
giorno dell'anno: 184
ora estiva: 1
```

604.4.4 Funzioni «gmtime()» e «localtime()»

Le funzioni **'gmtime()'** e **'localtime()'** hanno in comune il fatto di ricevere come argomento il puntatore di tipo **'time_t *'**, a un'informazione data-orario, per restituire il puntatore a una variabile strutturata di tipo **'struct tm *'**. In altri termini, le due funzioni convertono una data espressa nella forma del tipo **'time_t'**, in una data suddivisa nella struttura **'tm'**:

```
struct tm *gmtime    (const time_t *timer);
struct tm *localtime (const time_t *timer);
```

Nell'ambito di queste funzioni, è ragionevole supporre che l'informazione di tipo **'time_t'** a cui fanno riferimento, sia espressa in termini di tempo universale e che le funzioni stesse abbiano la possibilità di stabilire il fuso orario e la modalità di regolazione dell'ora estiva.

In ogni caso, la differenza tra le due funzioni sta nel fatto che **'gmtime()'** traduce il tempo a cui punta il suo argomento in una struttura contenente la data tradotta secondo il tempo coordinato universale, mentre **'localtime()'** la traduce secondo l'ora locale.

Va osservato che queste funzioni restituiscono un puntatore a un'area di memoria che può essere sovrascritta da altre chiamate alle stessi funzioni o a

funzioni simili.

604.5 Conversione in stringa

Un piccolo gruppo di funzioni del file `'time.h'` è destinato alla conversione dei valori data-orario in stringhe, per l'interpretazione umana.

604.5.1 Funzione «asctime()»

La funzione **`'asctime()'`** converte un'informazione data-orario, espressa nella forma di una struttura **`'struct tm'`**, in una stringa che esprime l'ora locale, usando però una rappresentazione fissa in lingua inglese:

```
char *asctime (const struct tm *timeptr);
```

In pratica, dal momento che la data e l'orario vanno espressi secondo le convenzioni della lingua inglese, lo standard stesso descrive completamente questa funzione e il listato seguente è tratto letteralmente da tale definizione:

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

604.5.2 Funzione «ctime()»

La funzione **`'ctime()'`** converte un'informazione data-orario, espressa nella forma del tipo **`'time_t'`** in una stringa che esprime l'ora locale, usando però una rappresentazione fissa in lingua inglese:

```
char *ctime (const time_t *timer);
```

Il comportamento di questa funzione è tale da generare una stringa analoga a quella della funzione **'asctime()'**, tanto che si la si potrebbe esprimere così:

```
char *ctime (const time_t *timer)
{
    return asctime (localtime (timer));
}
```

Oppure, come macro-istruzione, così:

```
#define ctime(t) (asctime (localtime (t)));
```

604.5.3 Funzione «strftime()»

La funzione **'strftime()'** si occupa di interpretare il contenuto di una struttura di tipo **'struct tm'** e di tradurlo in un testo, secondo una stringa di composizione libera. In altri termini, questa funzione si comporta in modo simile a **'printf()'**, dove l'input è costituito dalla struttura contenente le informazioni data-orario.

```
size_t strftime (char * restrict s,
                 size_t maxsize,
                 const char * restrict format,
                 const struct tm * restrict timeptr);
```

Dal modello del prototipo della funzione, si vede che questa restituisce un valore numerico di tipo **'size_t'**. Questo valore rappresenta la quantità di elementi[(3)] che sono stati scritti nella stringa di destinazione, rappresentata dal primo parametro. Dal computo di questi elementi è escluso il carattere nullo di terminazione, ma questo viene comunque aggiunto dalla funzione.

La funzione richiede, nell'ordine: un array di caratteri da utilizzare per comporre il testo; la dimensione massima di questo array; la stringa di composizione, contenente del testo costante e degli specificatori di conversione; il puntatore alla struttura contenente le informazioni data-orario da usare nella conversione.

La funzione termina il proprio lavoro con successo solo se può scrivere

nell'array di destinazione il testo composto secondo le indicazioni della stringa di composizione, includendo anche il carattere nullo di terminazione. Se ciò non avviene, il valore restituito dalla funzione è zero e il contenuto dell'array di destinazione è imprecisato.

Il listato successivo mostra un programma completo che dimostra il funzionamento di **'strftime()'**. Va osservato che la conversione eseguita da tale funzione è sensibile alla configurazione locale; precisamente dipende dalla categoria **'LC_TIME'**:

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

int
main (int argc, char *argv[])
{
    char s[100];
    time_t t      = time (NULL);
    struct tm *tp = localtime (&t);
    int dim;
    setlocale (LC_ALL, "it_IT.UTF-8");
    dim = strftime (s, 100, "Ciao amore: sono le %H:%M del %d %B %Y.", tp);
    printf ("%d: %s\n", dim, s);
    return 0;
}
```

Ecco cosa si potrebbe ottenere eseguendo questo programma:

```
45: Ciao amore: sono le 09:32 del 27 giugno 2007.
```

Nella tabella successiva vengono elencati gli specificatori di conversione principali. Sono ammissibili delle varianti, con l'aggiunta di modificatori, che però non vengono descritte. Per esempio è ammissibile l'uso degli specificatori **'%Ec'** e **'%Od'**, per indicare rispettivamente una variante di **'%c'** e **'%d'**.

*Tabella 604.20. Specificatori di conversione usati dalla funzione **'strftime()'**.*

Specificatore	Corrispondenza
%C	*century* Il secolo, ottenuto dividendo l'anno per 100 e ignorando i decimali.

Specificatore	Corrispondenza
`%y` `%Y`	*year* L'anno: nel primo caso si mostrano solo le ultime due cifre, mentre nel secondo si mostrano tutte.
`%b`|`%h` `%B`	Rispettivamente, il nome abbreviato e il nome per esteso del mese.
`%m`	*month* Il numero del mese, da 01 a 12.
`%d` `%e`	*day* Il giorno del mese, in forma numerica, da 1 a 31, utilizzando sempre due cifre: nel primo caso si aggiunge eventualmente uno zero; nel secondo si aggiunge eventualmente uno spazio.
`%a` `%A`	Rispettivamente, il nome abbreviato e il nome per esteso del giorno della settimana.
`%H` `%L`	*hour* L'ora, espressa rispettivamente a 24 ore e a 12 ore.
`%p`	La sigla da usare, secondo la configurazione locale, per specificare che si tratta di un'ora antimeridiana o pomeridiana. Nella convenzione inglese si ottengono, per esempio, le sigle «AM» e «PM».
`%r`	L'ora espressa a 12 ore, completa dell'indicazione se trattasi di ora antimeridiana o pomeridiana, secondo le convenzioni locali.
`%R`	L'ora e i minuti, equivalente a **`'%H:%M'`**.
`%M`	*minute* I minuti, da 00 a 59.
`%S`	*second* I secondi, espresso con valori da 00 a 60.
`%T`	*time* L'ora, i minuti e i secondi, equivalente a **`'%H:%M:%S'`**.

Specificatore	Corrispondenza
%z %Z	*time zone* La rappresentazione del fuso orario, nel primo caso come distanza dal tempo coordinato universale (UTC), mentre nel secondo si usa una rappresentazione conforme alla configurazione locale.
%j	*julian* Il giorno dell'anno, usando sempre tre cifre numeriche: da 001 a 366.
%g %G	L'anno a cui appartiene la settimana secondo lo standard ISO 8601: nel primo caso si mostrano solo le ultime due cifre, mentre nel secondo si ha l'anno per esteso. Secondo lo standard ISO 8601 la settimana inizia con lunedì e la prima settimana dell'anno è quella che include il 4 gennaio.
%V	Il numero della settimana secondo lo standard ISO 8601. I valori vanno da 01 a 53. Secondo lo standard ISO 8601 la settimana inizia con lunedì e la prima settimana dell'anno è quella che include il 4 gennaio.
%u %w	Il giorno della settimana, espresso in forma numerica, dove, rispettivamente, si conta da 1 a 7, oppure da 0 a 6. Zero e sette rappresentano la domenica; uno è il lunedì.
%U %W	Il numero della settimana, contando, rispettivamente, dalla prima domenica o dal primo lunedì di gennaio. Si ottengono cifre da 00 a 53.
%x	La data, rappresentata secondo le convenzioni locali.
%X	L'ora, rappresentata secondo le convenzioni locali.
%c	La data e l'ora, rappresentate secondo le convenzioni locali.
%D	*date* La data, rappresentata come **'%m/%d/%Y'**.
%F	La data, rappresentata come **'%Y-%m-%d'**.
%n	Viene rimpiazzato dal codice di interruzione di riga.

Specificatore	Corrispondenza
%t	Viene rimpiazzato da una tabulazione orizzontale.
%%	Viene rimpiazzato da un carattere di percentuale.

---

1) Il valore positivo massimo è (2^{31})-1, il quale, diviso per la quantità di secondi di un giorno (86 400) dà 24 855 che, diviso 365, dà circa 68 anni.

2) Il caso della funzione `'clock()'` e del tipo `'clock_t'` è stato considerato a parte.

3) Si tratta di byte: se il testo copiato è costituito da sequenze multibyte, i byte sono in quantità maggiore rispetto ai caratteri tipografici che si ottengono.

Dovrebbe essere possibile fare riferimento a questa pagina anche con il nome c_171_time_h_187.htm

[successivo] [precedente] [inizio] [fine] [indice generale] [indice ridotto] [indice analitico] [home]