Trabalhar com tempo!

Como utilizar algumas das funcionalidades oferecidas pela biblioteca time.h

António Manuel de Sousa Barros
Departamento de Eng. Informática
ISEP – Instituto Superior de Engenharia do Porto
Maio de 2003

Que tipos suportam dados temporais?

A biblioteca time.h oferece dois tipos de dados que suportam informação de calendário:

- time_t ,
- struct tm.

O tipo *time_t* guarda a informação de calendário (data + hora) num único número que representa univocamente um instante. Este número representa o total de segundos que passaram desde a meia-noite de 1 de Janeiro de 1970 (a data ZERO para os sistemas UNIX). Este tipo é óptimo para comparação entre datas, pois uma data posterior é representada por um número maior.

O tipo *struct tm* é apenas utilizado para representação de um instante num formato humanamente inteligível. Neste tipo, os dados sobre um instante estão divididos por classes (dia, mês, ano, hora, data, etc). Este tipo encontra-se definido em time.h da seguinte forma:

```
struct tm {
  int tm_sec;       /* segundos [0; 59] */
  int tm_min;       /* minutos [0; 59] */
  int tm_hour;      /* hours [0; 23] */
  int tm_mday;      /* dia do mês [1; 31] */
  int tm_mon;       /* mês (meses desde Janeiro) [0; 11] */
  int tm_year;      /* ano (desde 1900) [0; ...] */
  int tm_wday;      /* dia da semana (desde Domingo) [0; 6] */
  int tm_yday;      /* dia do ano (desde 1 Janeiro) [0; 365] */
  int tm_isdst;     /* informação sobre mudança da hora */
};
```

Como obtenho a informação horária deste instante?

A função *time()* vai buscar a informação ao relógio do sistema e retorna-a no formato *time_t*. O protótipo da função é o seguinte:

```
time_t time(time_t *t);
```

Exemplo:

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

main()
{
  time_t tempo;
  (...)
  time(&tempo); /* 'tempo' fica com a marca deste instante. */
  printf("%ld segundos\n", (long) tempo);
}
```

O resultado do exemplo anterior não é intelígivel...

Podemos transformar o conteúdo de uma variável do tipo *time_t* numa variável do tipo *struct tm*, utilizando a função *localtime()*, cujo protótipo é:

```
struct tm *localtime(const time *timep);
```

Exemplo:

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

main()
{
  time_t tempo;
  struct tm *data;

  time(&tempo); /* 'tempo' fica com a marca deste instante. */
  printf("%ld segundos\n", (long) tempo);

  data = localtime(&tempo);

  printf("DATA:\n");
  printf("Ano: %d\n", (*data).tm_year+1900);
  printf("Mês: %d\n", (*data).tm_mon+1);
  printf("Dia: %d\n", (*data).tm_mday);

  printf("HORAS:\n");
  printf("Horas: %d\n", (*data).tm_hour);
  printf("Minutos: %d\n", (*data).tm_min);
  printf("Segundos: %d\n", (*data).tm_sec);
}
```

Mas agora eu quero fazer o contrário...

Para passar de *struct tm* para *time_t*, utiliza-se a função *mktime()*, cujo protótipo é:

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

Exemplo:

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

main()
{
  time_t tempo;
  struct tm data;

  printf("DATA:\n");
  printf("Ano: ");
  scanf("%d", &data.tm_year);
  data.tm_year -= 1900;    /* Retira o offset de 1900. */
  printf("Mês: ");
  scanf("%d", &data.tm_mon);
```

```
  data.tm_mon--;           /* Retira o offset de 1. */
  printf("Dia: ");
  scanf("%d", &data.tm_mday);

  printf("HORAS:\n");
  printf("Horas: ");
  scanf("%d", &data.tm_hour);
  printf("Minutos: ");
  scanf("%d", &data.tm_min);
  printf("Segundos: ");
  scanf("%d", &data.tm_sec);

  tempo = mktime(&data);

  printf("Tempo: %ld\n", (long) tempo);

  if(tempo < time(NULL))
    printf("Essa data já ocorreu!\n\n");
  else
    printf("Essa data ainda não ocorreu!\n\n");
}
```