Tempo e Data em Linux

1- Introdução:

A manipulação de variáveis de tempo externo é fundamental em programação para tempo real. Deste modo, torna-se básico o conhecimento de como se obter variáveis de tempo e data em linux. Este conhecimento pode ser aplicado a uma infinidade de situações, como: medição da duração de um determinado programa para efeito de avaliação de desempenho e programação de disparo tarefa.

Todos os sistemas Unix-like, que é o caso do Linux/GNU, utilizam o mesmo ponto de partida para medição de tempo e data: meia-noite de GMT de primeiro de janeiro de 1970. Assim, todos os tempos em Unix são medidos em segundos a partir dessa data.

A variável Tempo é manipulada utilizando-se o tipo pré-definido **time_t**. Sendo este do é um tipo inteiro longo (**long**), de modo a poder conter valores de data e tempo em segundos. Esse tipo é definido no arquivo de cabeçalho **time.h**.

A função **time(time_t *ptr_tempo)** retorna o tempo em segundo transcorrido desde a data de referência. Este valor também será escrito na posição de memória apontada pelo ponteiro ptr_tempo:

```
#include <time.h>
time_t time(time_t *ptr_tempo);
```

O programa tempo1.cpp ilustra a utilização desta função.

```
// UFRN-CT-DCA, Primeiro semestre de 2003
// Programação para Tempo Real
// Programa: tempo1
#include <iostream> // para: cout
#include <time.h>   // para: time()
#include <unistd.h>
using std::cout;
int main ( )
{
 time_t   tempo_real;
 tempo_real   =  time( (time_t *) 0);
 cout << "Já se passaram  " << tempo_real << " segundos desde 0:00:00 de 01/01/1970 " << '\n';
 exit(0);
}
```

Questão 1) Estenda *Programa1* para após escrever o valor do tempo atual, ele leia um número inteiro do teclado e depois volta a escrever o novo valor de tempo.

Uma outra função muito importante é a **sleep(time)**, que suspende a execução do programa por **time** segundos. O Programa *tempo1.cpp* ilustra a utilização desta função.

```
// UFRN-CT-DCA, Primeiro semestre de 2003
// Programação para Tempo Real
// Programa: tempo2
```

```
#include <iostream> // para: cout
#include <stdio.h>
#include <unistd.h> // para: sleep()
using std::cout;
main ( )
{
  double x = 1;
  for(int i=0; i<5; i++) {
    sleep(i);
   cout << x +i << '\n';
  }
  //   std::cout << x <<'\n';
}
```

Questão 2) Faça um programa que leia e escreva o valor do tempo atual de 2 em 2 segundos durante 20 segundos.

Como poderíamos utilizar a função time para calcular o tempo vasto numa determinada atividade?

- Uma possível solução seria obter o **tempo** real antes e depois da atividade e então calcular a diferença deles!
- O padrão ISAO/ANSI definiu uma função **difftime** para calcular esta diferença.

```
#include <time.h>
double difftime(time_t tempo1, time_t tempo2)
```

A função retorna um **double** contendo a diferença entre tempo1 e tempo2 (tempo1 – tempo2).

Questão 3) Reescreva o programa da Questão 1 para que o mesmo escreva o tempo que o usuário gasta para fornecer o número solicitado.

Para se apresentar o tempo em forma mais amigável ao usuário, há funções padrões de conversão. Uma dela é a função **gmtime:**

```
#include <time.h>
struct tm *gmtime (const time_t tempo);
```

No caso, essa função retorna um ponteiro para um estrutura que contém os seguintes campos:

```
int tm_sec;          // segundos, 0-61
int tm_min;          // minutos, 0-59
int tm_hour;         // horas, 0-23
int tm_mday;         // dia do mês, 1-31
int tm_mon;          // mês no ano, 0-11
int tm_year;         // ano desde 1900
int tm_wday;         // dia na semana, 0-6
int tm_yday;         // dia no ano, 0-365
```

O programa tempo3.cpp ilustra a utilização da função gmtime().

```
#include <iostream>
#include <time.h>
#include <unistd.h> // para: sleep()
using std::cout;

int main ( )
{

  struct tm *tm_ptr; // um ponteiro para a estrutura tm
  time_t  tempo_valor;  // variável para armazenar o tempo em segundos

  (void) time(&tempo_valor);  // outra forma de armazenar o tempo em segundos
  tm_ptr = gmtime(&tempo_valor); // obtendo a esttutura

  cout << "O tempo em segundos é;  " << tempo_valor << '\n';
  cout << "Saida padronizada " << '\n';
  cout << "Data: " << tm_ptr->tm_mday << "/"
                          <<tm_ptr->tm_mon +1 << "/" <<1900 + (tm_ptr->tm_year) << '\n';
}
```