

Obtendo tempo de execução de código

Antes de iniciar, é importante definir corretamente alguns termos relacionados a tempo:

- Tempo absoluto (`calendar time`) é um instante preciso na referência de tempo universal, como 18/10/2004, 14:00:00.000 GMT-3. Uma data é parte do tempo absoluto.
- Intervalo é uma parte contínua de tempo entre dois tempos absolutos. Um exemplo seria o intervalo de 13:30:00 a 17:50:00 de 18/10/2004.
- Tempo decorrido (`elapsed time`) é o tamanho de um intervalo. No exemplo acima, o tempo decorrido seria de 4:20:00.
- A duração (`amount of time`) é a soma dos tempos decorridos em todos os intervalos considerados. Por exemplo, a duração de nosso curso é de 40 horas, distribuídas entre vários intervalos.
- O período é o tempo decorrido no intervalo entre dois eventos, considerado sobretudo quando esses eventos são parte de uma sequência de eventos repetitivos.
- O tempo de CPU (`CPU time`) é similar ao tempo absoluto, mas considerado para cada processo em particular.
- O tempo de processamento (`processor time`) é a duração de uso da CPU por um ou mais processos.

Tempo absoluto

Várias funções permitem consultar o tempo absoluto do sistema ou manipular variáveis contendo informações de tempo absoluto. Várias granularidades de tempo são possíveis, do segundo ao micro-segundo.

Tempo absoluto simples

O tipo `time_t` pode representar o tempo absoluto ou tempos decorridos, e pertence a biblioteca `time.h`, seu protótipo é apresentado abaixo:

```
time_t time (time_t *result);
```

A função recebe um ponteiro para um tipo `time_t` e devolve um valor do tipo `time_t`. Quando usado para representar o tempo absoluto, indica o número de segundos decorridos desde 01/01/1970 00:00:00 UTC (esse instante é conhecido como *epoch*). Fusos horários não são considerados.

A função `difftime()`, disponível na biblioteca `time.h`, retorna o número de segundos decorridos entre os dois tempos absolutos informados. Essa é a única forma portátil

de fazer esse cálculo, pois a implementação do tipo `time_t` pode mudar entre sistemas. Um pequeno exemplo do uso das funções `time` e `difftime()`:

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

int main(){
    time_t antes, depois;

    antes = time(0); //obtem o tempo neste instante
    Sleep(2);        //gera delay - atraso
    depois = time(0); //obtem o tempo neste instante

    printf("Entre %d e %d se passaram %f segundos\n",
           antes, depois, difftime(depois, antes));
}
```

Figura 1 - exemplo das funções `time()` e `difftime()`

Tela de execução

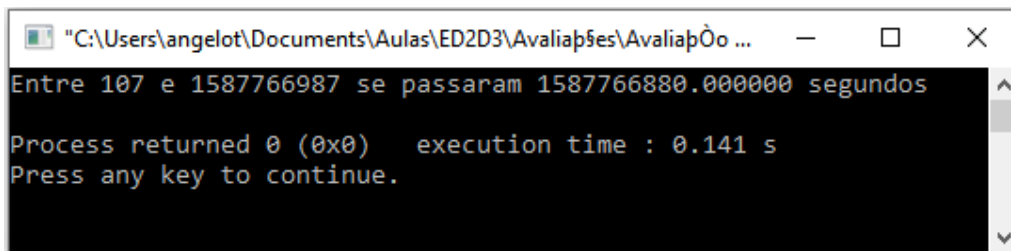


Figura 2 - Execução das funções `time()` e `difftime()`

Tempo absoluto preciso

A função `gettimeofday()` permite obter tempos absolutos com mais resolução que as apresentadas anteriormente. Ela permite informar o tempo absoluto do sistema, desde o evento *epoch*, utilizando para sua representação uma struct chamada `timeval`. Ambas, a função `gettimeofday()` e a struct `timeval` pertencem à biblioteca `sys/time.h`, e seus protótipo e declaração são:

```
struct timeval{
    long int tv_sec; // segundo decorridos
    long int tv_usec; // micro-segundos decorridos
} // (valor mínimo depende do sistema)

int gettimeofday (struct timeval *tp, struct timezone *tzp);
```

A função `gettimeofday()` recebe como parâmetros, um ponteiro para uma struct `timeval`, onde os valores de tempo em segundos do relógio absoluto serão armazenado através do ponteiro do primeiro argumento, e como segundo argumento, o fuso horário local que é informado através de ponteiro para struct `timezone` `*tzp`. Nos sistemas atuais esse ponteiro deve ser nulo, pois essa é uma característica obsoleta do UNIX 4.3 BSD mantida nas funções para preservar a compatibilidade. A função devolve zero em caso de sucesso e -1 em caso de erro. A seguir um programa exemplo de utilização da função `gettimeofday()`:

```
#include <stdio.h>
#include <stdlib.h>
#include <sys/time.h>

int main () {
    struct timeval Tempo_antes, Tempo_depois;
    double deltaT;

    gettimeofday (&Tempo_antes, NULL); //coleta time no inicio
    //entre estas duas chamadas de gettimeofday() que será medido
    //o intervalo de tempo em segundos gasto para imprimir em tela
    printf("Esta mensagem demora quanto tempo\n");
    printf("para ser impressa em tela?\n\n");

    gettimeofday (&Tempo_depois, NULL); //coleta time no final

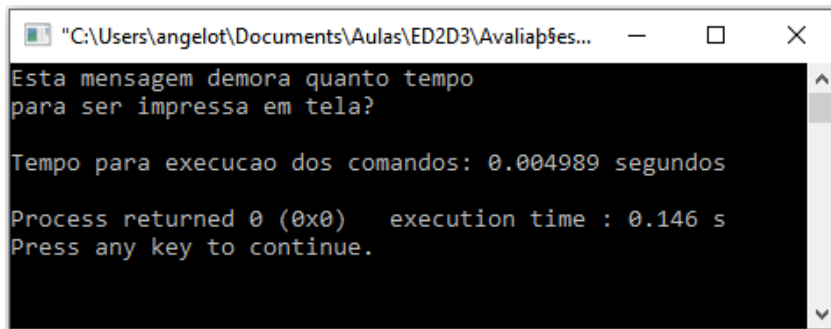
    deltaT = (Tempo_depois.tv_sec + Tempo_depois.tv_usec/1000000.0) -
             (Tempo_antes.tv_sec + Tempo_antes.tv_usec /1000000.0);
    printf ("Tempo para execucao dos comandos: %f segundos\n", deltaT);
    //exibe o tempo gasto para imprimir em tela os dois comandos printf
    //ou seja, o intervalo entre a hora no inicio e no final, subtraídos.
}
```

Figura 3 - Utilização da função `gettimeofday()`

Note que a variável `deltaT` recebe a diferença, ou seja, o intervalo de tempo entre os valores em segundos e micro-segundos armazenados respectivamente em `Tempo_depois.tv_sec + Tempo_depois.tv_usec`, menos os segundos e micro-segundo armazenados em `Tempo_antes.tv_sec + Tempo_antes.tv_usec`.

A divisão dos campos `Tempo_antes.tv_usec` e `Tempo_depois.tv_usec`, por 1.000.000 apresenta os valores em segundos com precisão de microssegundos nas casas decimais. Analogamente se a divisão for por 1.000, a precisão se dará com os milissegundos nas casas decimais, e assim por diante. Os resultados sempre serão totalizados em segundos. Caso seja preciso a representação em horas ou minutos, faz-se necessária uma prévia etapa de conversão.

Tela de execução



```
"C:\Users\angelot\Documents\Aulas\ED2D3\Avaliapes..."
Esta mensagem demora quanto tempo
para ser impressa em tela?

Tempo para execucao dos comandos: 0.004989 segundos

Process returned 0 (0x0)  execution time : 0.146 s
Press any key to continue.
```

Figura 4 - Execução de `gettimeofday()`

É possível observar na tela de execução que o programa apresentado na figura 3 gastou 0,004989 segundos para executar os dois comandos `printf()` com as mensagens, ou ainda 4 milisegundos e 989 microsegundos, este é o nível de precisão.