

Функции замера времени



Назначение функций замера времени

Функции замера времени
необходимы для оценки
эффективности работы
алгоритма, процессора, памяти,
файловой системы, сети ...

Методика тестирования

- Фиксирование времени начала теста
- Запуск тестируемой операции
- Фиксирование времени окончания теста
- Отображение результатов теста

Функция clock

- Библиотечная функция определена в заголовочном файле `time.h`
- Прототип `clock_t clock();`
- Возвращает время, прошедшее с момента запуска программы в единицах `1/CLK_TCK` секунды
- Используется в Windows

Пример использования функции clock

```
#include <stdio.h>
#include <time.h>
int main()
{
    double pi;
    clock_t t;
    t=clock(); // начало замера времени
    pi = pi_calculate();
    t=clock()-t; // окончание замера времени
    printf("Time: %lf msec Pi = %lf\n",t*1000.0/CLK_TCK,pi);
    return 0;
}
```

Достоинства и недостатки функции clock

- Достоинства:
 - ✓ высокая платформенная независимость
- Недостатки:
 - ✓ низкая точность, при высокой загрузке процессора – неприемлемая точность, так как измеряется интервал времени, во время которого помимо процесса исследуемой программы исполнялись и другие процессы

Команда RDTSC

- Платформенно-зависимый вариант для x86
- Возвращает число тактов с момента запуска процессора
- Используется в Windows и UNIX для процессоров Intel

Пример использования команды RDTSC

```
#include<stdio.h>
#define FREQ 1995 // Частота процессора, MHz

// функция возвращает значение счетчика тактов процессора
unsigned long tick()
{
    __asm rdtsc
}

int main(int argc, char* argv[])
{
    unsigned long t1,t2;
    double t, pi;
    t1=tick(); // начало замера
    pi = pi_calculate();
    t2=tick(); // конец замера
    t=(double)(t2-t1)/FREQ;
    printf("Time: %ld mcsec Pi = %lf\n",t,pi);
    return 0;
}
```


Достоинства и недостатки команды RDTSC

- Достоинства:
 - ✓ максимально возможная точность
- Недостатки:
 - ✓ зависимость от архитектуры процессора
 - ✓ ухудшение точности при высокой загрузке процессора

Функция `gettimeofday`

- Библиотечная функция определена в заголовочном файле `sys/time.h`
- Прототип
`gettimeofday(struct timeval* tv, struct timezone* tz);`
- Время можно вычислить из структуры `timeval`
- Используется в UNIX

int

Пример использования функции gettimeofday

```
#include <sys/time.h>
struct timeval tv1, tv2, dtv;
struct timezone tz;
void time_start() { gettimeofday(&tv1, &tz); }
double time_stop()
{
    gettimeofday(&tv2, &tz);
    dtv.tv_sec = tv2.tv_sec - tv1.tv_sec;
    dtv.tv_usec = tv2.tv_usec - tv1.tv_usec;
    if(dtv.tv_usec < 0) { dtv.tv_sec--; dtv.tv_usec += 1000000; }
    return dtv.tv_sec * 1000.0 + dtv.tv_usec / 1000.0;
}
```

Пример использования функции gettimeofday

```
#include <stdio.h>
int main()
{
    double pi;
    time_start(); // начало замера времени
    pi = pi_calculate();
    // окончание замера времени
    printf("Time: %lf msec Pi = %lf\n",time_stop(),pi);
    return 0;
}
```

Достоинства и недостатки функции `gettimeofday`

- Достоинства:
 - ✓ высокая платформенная независимость
- Недостатки:
 - ✓ низкая точность, при высокой загрузке процессора – неприемлемая точность, так как измеряется интервал времени, во время которого помимо процесса исследуемой программы исполнялись и другие процессы

Функция `times`

- Библиотечная функция определена в заголовочном файле `sys/times.h`
- Прототип `clock_t times(struct tms *buf);`
- Возвращает время, прошедшее с момента запуска программы в единицах `1/CLK_TCK` секунды
- Используется в UNIX

Пример использования функции times

```
#include <sys/times.h>
#include <time.h>
struct tms tmsBegin,tmsEnd;
void time_start() { times(&tmsBegin); }
double time_stop()
{ times(&tmsEnd);
  return ((tmsEnd.tms_ftime-tmsBegin.tms_ftime)+
          (tmsEnd.tms_stime-tmsBegin.tms_stime))*1000.0/CLK_TCK;
}
```

Пример использования функции times

```
#include <stdio.h>
int main()
{
    double pi;
    time_start(); // начало замера времени
    pi = pi_calculate();
    // окончание замера времени
    printf("Time: %lf msec Pi = %lf\n",time_stop(),pi);
    return 0;
}
```


Достоинства и недостатки функции times

- Достоинства:
 - ✓ высокая точность (относительная независимость от других процессов системы)
- Недостатки:
 - ✓ для малых интервалов она зависит от интервала времени прерываний по таймеру