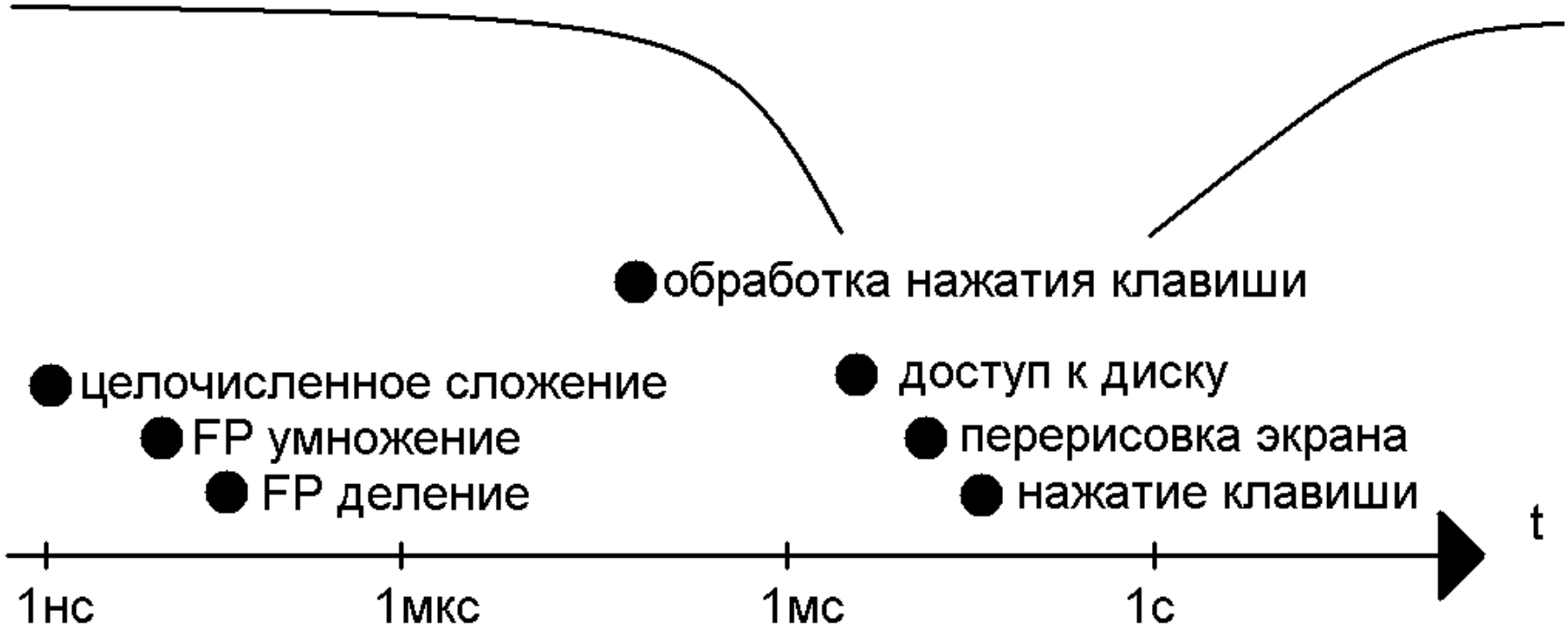
НГТУ
Кафедра параллельного программирования

Измерение времени в ЭВМ

С.Е. Киреев, В.П. Маркова,
М.Б. Остапкевич, В.А. Перепелкин

Новосибирск
2013

Временная шкала событий в ЭВМ



(для системы с тактовой частотой 1 ГГц)

Аппаратные ресурсы для измерения времени

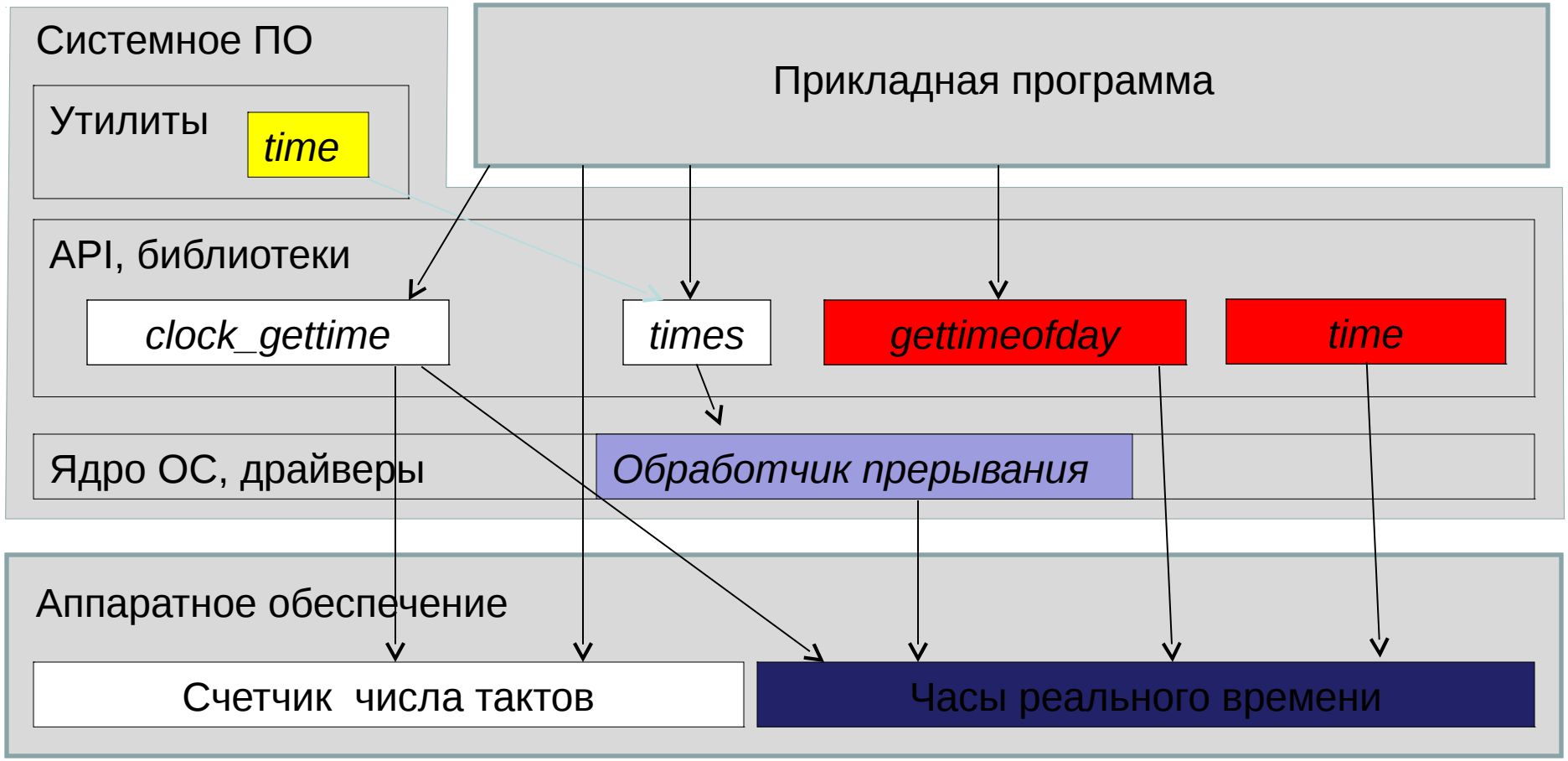
- Счетчик числа тактов процессора

  (точный для измерения малых промежутков времени до 10 мсек., зависит от архитектуры, есть не для всех архитектур, в SMP нужна привязка процессов)

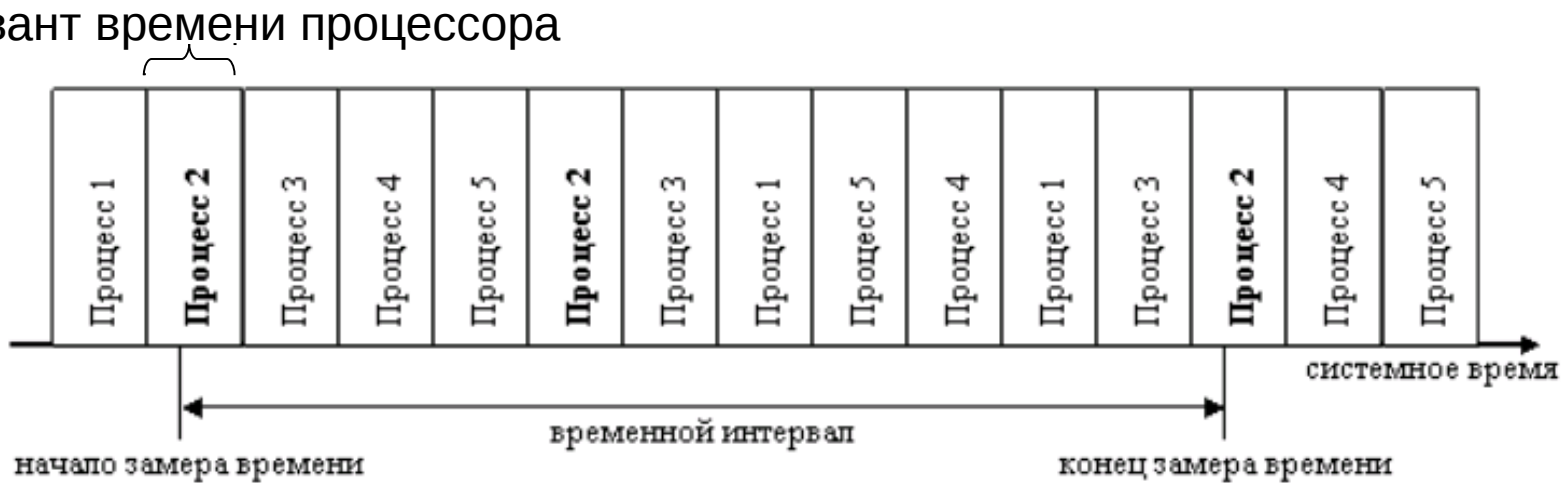
- Часы реального времени

  (удовлетворительная точность для промежутков времени более 1 сек.)

Основные способы измерения времени в ОС Linux



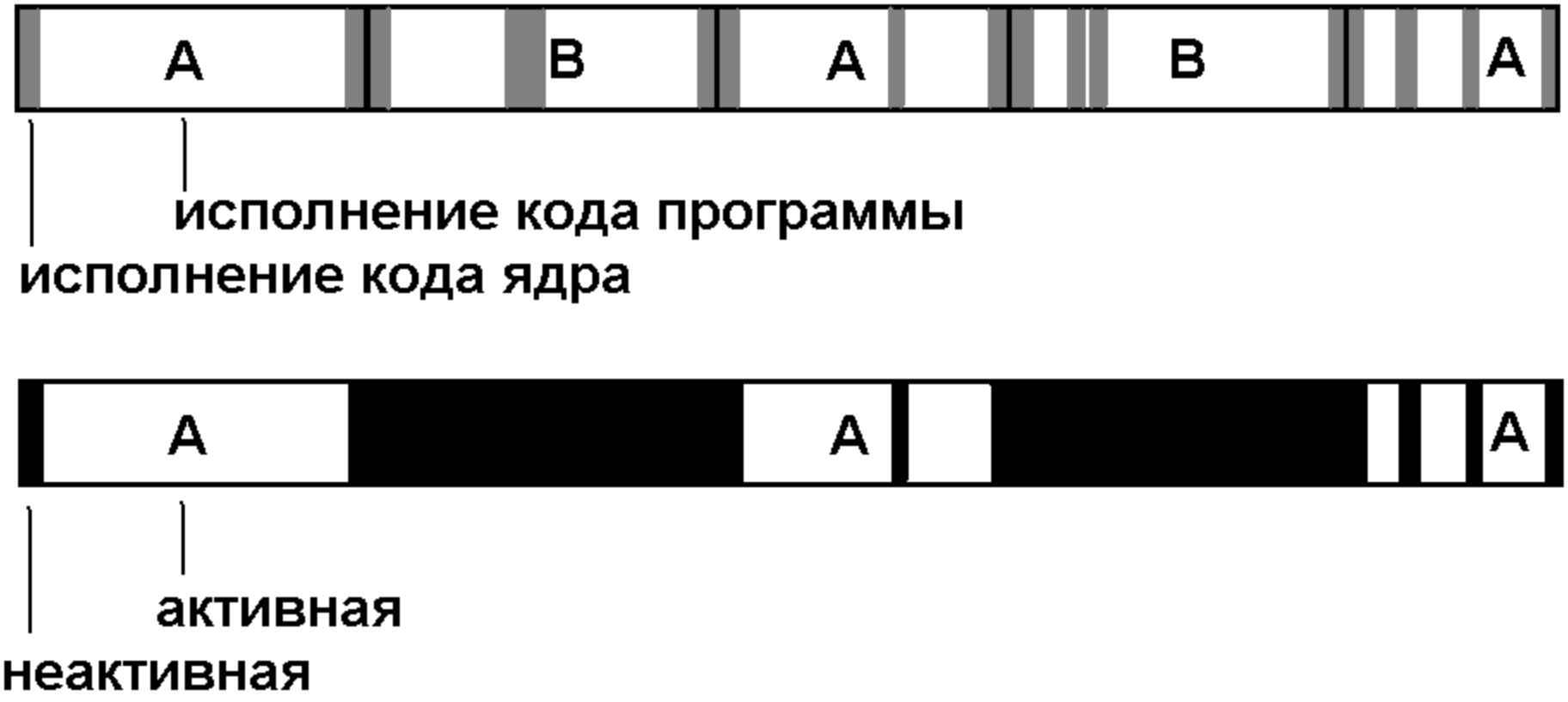
Измерение временных интервалов в языке Си



Измерение времени работы программы в многозадачной операционной системе (Windows, Linux, …) имеет свои особенности. В такой системе каждый процессор (ядро) всегда выполняет несколько процессов (программ) в режиме разделения времени.

Операционная система выделяет каждому процессу квант времени процессора и управляет переключением процессора с одного процесса на другой. Таким образом, если замерить время работы некоторой программы внешним хронометром, отражающим реальное течение времени, то этот интервал попадет и время работы каких-то других процессов.

Измерение времени в системах с разделением времени



Измерение временных интервалов в языке Си

Все дальнейшие примеры использования функций замера интервалов времени будут основаны на функциях
1. `void time_start()`, вызываемую в момент начала замера времени;
2. `long time_stop()`, возвращающую количество миллисекунд, прошедших с начала замера времени.

Пример использования функций замера времени:

```
main()
{
    . . .
    time_start();
    /* вычисления */
    printf(“Время: %ld миллисекунд.\n", time_stop());
    . . .
}
```

Измерение системного времени

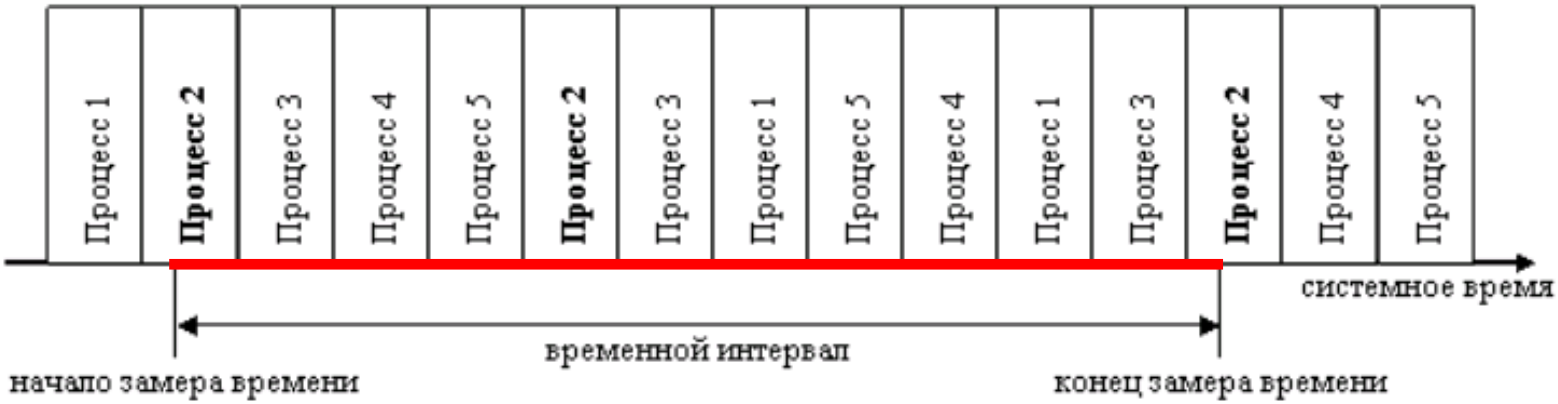
Счетчик системного времени

Вычислительная система имеет несколько программных и аппаратных счетчиков, отражающих течение времени с различных точек зрения. Необходимо различать следующие счетчики:

Счетчик системного времени (system time, wall-clock time) – программный счетчик, который отражает течение времени с точки зрения операционной системы и, как правило, соответствует реальному течению времени. Значение системного времени в каждый момент одинаково для всех программ, работающих на данном компьютере.

Функции для измерения системного времени:

- Windows: `GetSystemTime(), GetTickCount(), time(), clock(),`
- Linux: `gettimeofday(), time(), clock().`



Измерение системного времени

Функция
`int gettimeofday(struct timeval *tv, struct timezone *tz)`
возвращает в полях `tv_sec` и `tv_usec` переменной `tv` количество секунд и микросекунд, прошедших с полуночи 1 января 1970 года.

Пример использования:

```
#include <sys/time.h>
struct timeval tv1, tv2, dtv;
struct timezone tz;
//функция для начала замера времени, сохраняющая в переменной tv1 время начала измерений
void time_start()
{ gettimeofday(&tv1, &tz); }
//функция для окончания замера времени, возвращающая количество миллисекунд, прошедших с начала замера времени
long time_stop()
{ gettimeofday(&tv2, &tz);
  dtv.tv_sec= tv2.tv_sec  - tv1.tv_sec;   //разница секунд
  dtv.tv_usec=tv2.tv_usec - tv1.tv_usec;  //разница микросекунд
  if (dtv.tv_usec<0) { dtv.tv_sec--; dtv.tv_usec+=1000000; }
  //возвращение количества прошедших миллисекунд
  return dtv.tv_sec*1000 + dtv.tv_usec/1000;
}
```

Измерение системного времени

Функция
`clock_t clock()`
возвращает количество тактов, прошедших с момента запуска программы

```
#include <time.h>
clock_t tm;
void time_start() { tm = clock(); }
long time_stop()
{
    return (clock() – tm)*1000/CLOCKS_PER_SEC;
}
```

`CLOCKS_PER_SEC` – константа, значение которой равно количеству тактов исполняемых за секунду. Таким образом,
`(clock() – tm)*1000/CLOCKS_PER_SEC` – количество миллисекунд, прошедших с «момента времени» tm.

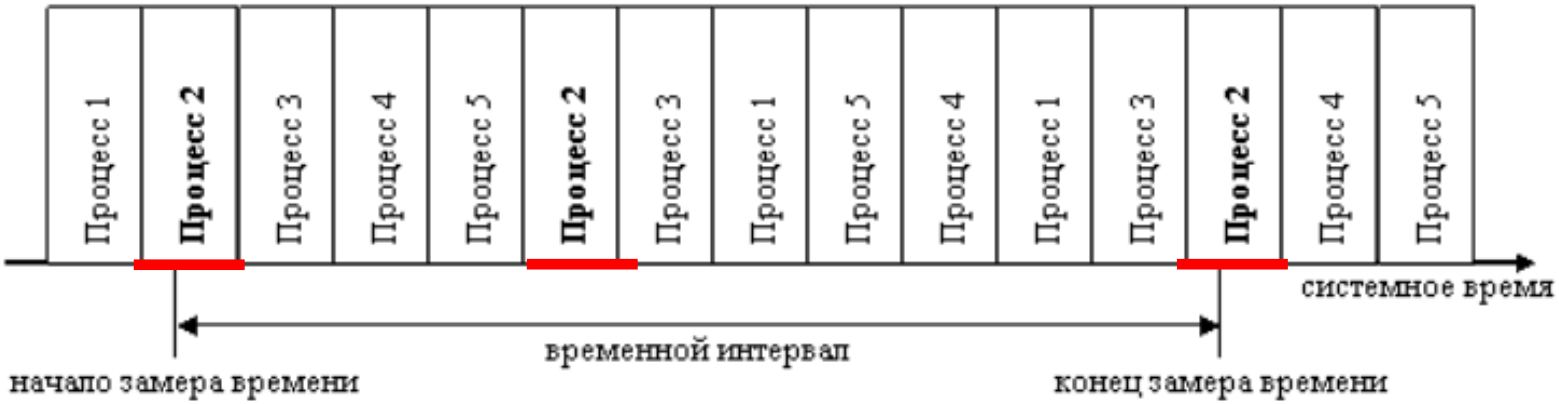
Измерение времени процесса

Счетчик времени процесса

Счетчик времени процесса (process time, CPU time) – программный счетчик, который отражает использование процессорного времени только конкретным процессом. Шаг изменения этого счетчика относительно велик, поэтому его не следует использовать для измерения малых промежутков времени.

Функции для получения времени процесса:

- Windows: `GetThreadTimes()`,`GetProcessTimes()`,
- Linux: `times()`.



Измерение времени процесса

Функция
`clock_t times (tms *buffer)`
Возвращает в поле `tms_utime` переменной `buffer` количество тактов, потраченных на исполнение инструкций пользовательского процесса с момента начала его исполнения; в поле `tms_stime` переменной `buffer` количество тактов, затраченных на исполнение системных вызовов инициированных процессом.

Использование функции `times`. Перевод тактов в миллисекунды производится так же, как и в примере для функции `clock`.

```
#include <sys/times.h>
#include <time.h>
struct tms tmsBegin,tmsEnd;
void time_start() { times(&tmsBegin); }
long time_stop()
{ times(&tmsEnd);
  return ((tmsEnd.tms_utime - tmsBegin.tms_utime)+
          (tmsEnd.tms_stime - tmsBegin.tms_stime))*
       1000/CLOCKS_PER_SEC;
}
```

Измерение времени процесса

Счетчик тактов процессора

Счетчик тактов процессора (CPU time stamp counter) – аппаратный счетчик, значение которого увеличивается на каждом такте процессора. Такт процессора – самый малый интервал времени в вычислительной системе, который теоретически может быть замерен. Поэтому счетчик тактов позволяет с большой точностью измерять малые промежутки времени (вплоть до нескольких команд процессора).

Счетчик тактов процессора имеет смысл использовать только для измерения интервалов времени меньших кванта времени, выделяемого процессу операционной системой. Для получения значения счетчика тактов используются специальные команды процессора, свои для каждой архитектуры:

- x86/x86-64: `rdtsc` (Read Time Stamp Counter)
- Alpha: `rpcc`,
- Itanium: `ar.itc`,
- PowerPC: `mftb`, `mftbu`.

Счетчик тактов процессора

`rdtsc` - ассемблерная инструкция для платформы x86, читающая счётчик TSC (Time Stamp Counter) и возвращающая его в регистрах EDX:EAX 64-битное количество тактов с момента последнего сброса процессора.

Пример использования инструкции `rdtsc` в ОС Linux:

```
#include <time.h>
long long TimeValue=0;
//функция, возвращающая количество тактов, прошедших с момента последнего сброса процессора
unsigned long long time_RDTSC()
{ union ticks
  { unsigned long long tx;
    struct dblword { long tl,th; } dw;
  } t;
  asm("rdtsc\n": "=a"(t.dw.tl),"=d"(t.dw.th));
  return t.tx;
}
void time_start() { TimeValue=time_RDTSC(); }
long long time_stop() {
       return (time_RDTSC()-TimeValue)*1000/CLOCKS_PER_SEC;
}
```

Счетчик тактов процессора

Пример использования инструкции `rdtsc` в Windows, MS Visual C++:

```
#include <intrin.h>
unsigned __int64 TimeValue=0;

unsigned __int64 rdtsc(void)
{
   return __rdtsc();
}

void time_start() { TimeValue=rdtsc(); }
long long time_stop() {
       return (rdtsc()-TimeValue)*1000/CLOCKS_PER_SEC;
}
```

```
int main( int argc, char **argv ){
   struct timespec start, stop;    double accum;

   if( clock_gettime( CLOCK_REALTIME, &start) == -1 ) {
     perror( "clock gettime" ); exit( EXIT_FAILURE );
   }

   system( argv[1] );

   if( clock_gettime( CLOCK_REALTIME, &stop) == -1 ) {
     perror( "clock gettime" ); exit( EXIT_FAILURE );
   }

   accum = ( stop.tv_sec - start.tv_sec )  +
         ( stop.tv_nsec - start.tv_nsec ) / BILLION;
   printf( "%lf\n", accum );
   return( EXIT_SUCCESS );
 }
```

Идентификатор таймера в clock_gettime

CLOCK_REALTIME

System-wide realtime clock. Setting this clock requires appropriate privileges.

CLOCK_MONOTONIC

Clock that cannot be set and represents monotonic time since some unspecified starting point.

CLOCK_PROCESS_CPUTIME_ID

High-resolution per-process timer from the CPU.

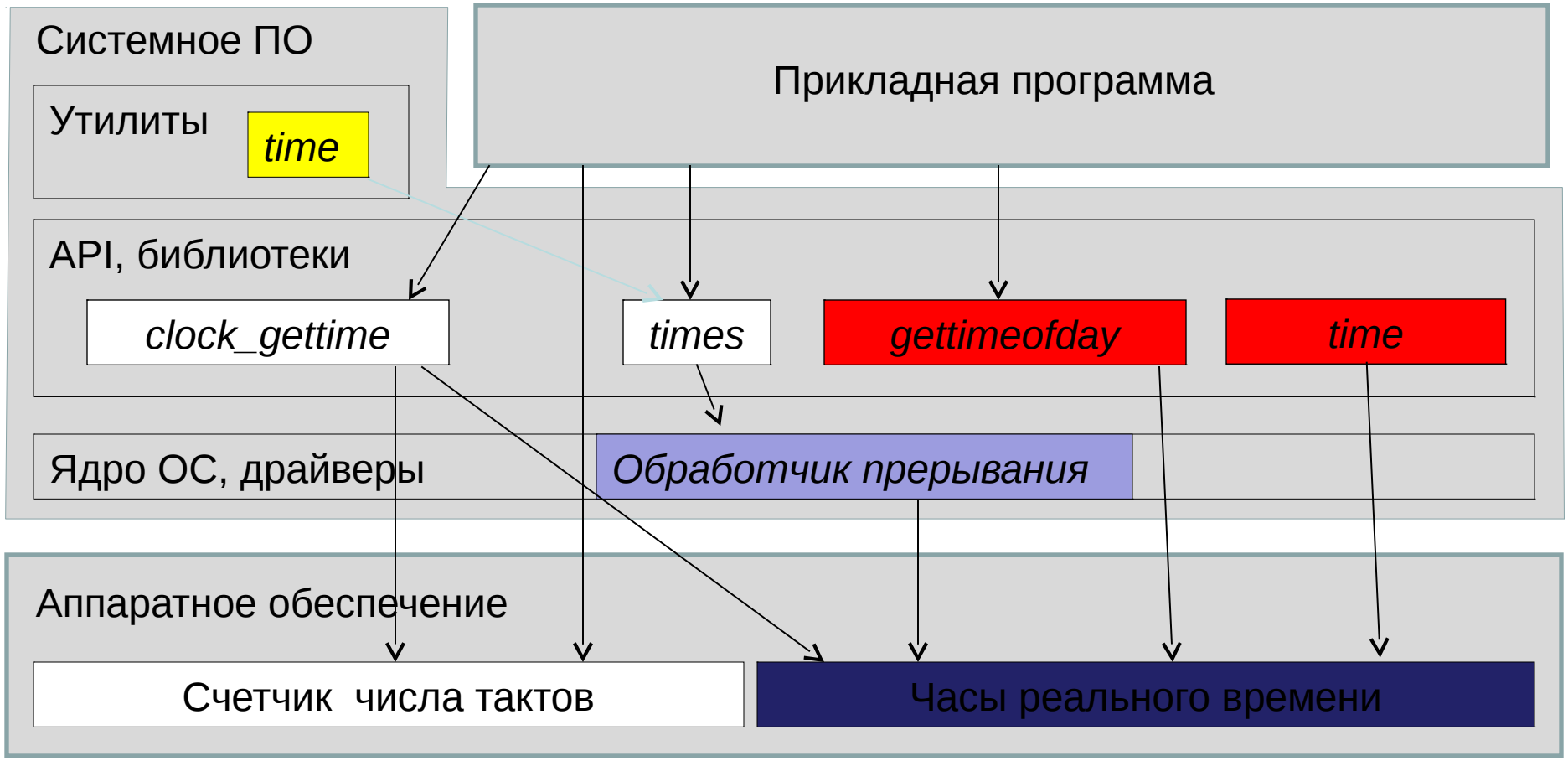
CLOCK_THREAD_CPUTIME_ID

Thread-specific CPU-time clock.

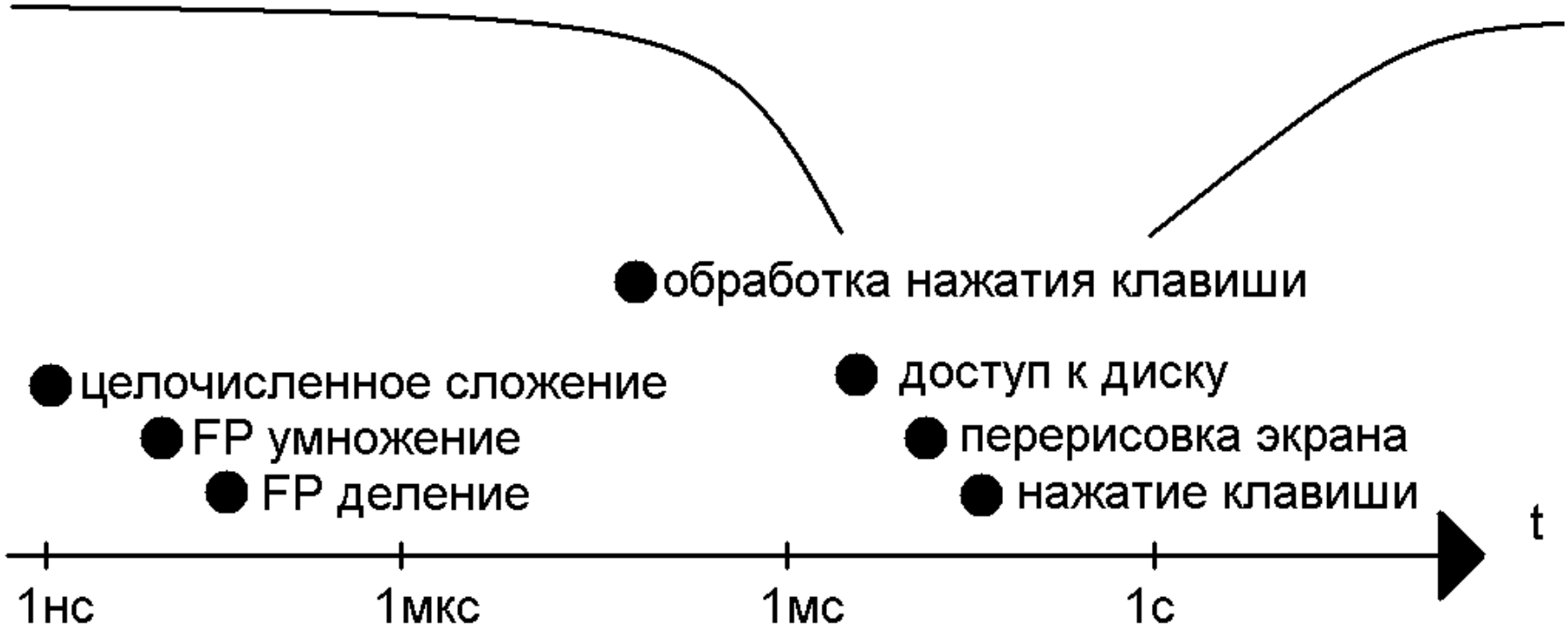
Факторы, вносящие искажения в измерение времени

- Другие процессы в многозадачных операционных системах
  - остановить другие приложения, оставить активными только необходимые сервисы или демоны ОС,
  - сделать несколько замеров, взять минимальное значение.
- Виртуальная память, дисковый кэш
  - запускать сброс дискового кэша (sync в Linux) перед каждым запуском программы,
  - сделать несколько замеров, взять минимальное значение.
- Разрешающая способность способа измерения времени
  - Подобрать способ в соответствии с априорной оценкой продолжительности.

Основные способы измерения времени в ОС Linux



Временная шкала событий в ЭВМ



(для системы с тактовой частотой 1 ГГц)