Subject: Problem with GetTickCount on Linux
Posted by steffen on Wed, 30 May 2012 09:38:31 GMT
View Forum Message <> Reply to Message

Hi,
On my embedded system I'm using a GPS to correct the system time and I found that adjusting the realtime clock on a posix system, cripples all the timer functions in Upp.

The Posix standard has added some clock_xxx functions to handle timers.
This gives a much better solution than the gettimeofday call used today.

Here is an updated function with fallback to the current gettimeofday call.

In Core/Util.cpp:

```
#ifdef PLATFORM_POSIX
dword GetTickCount() {
#if _POSIX_C_SOURCE >= 199309L
 struct timespec tp;
 if (clock_gettime(CLOCK_MONOTONIC, &tp) == 0)
 {
  return (dword)((tp.tv_sec * 1000) + (tp.tv_nsec / 1000000));
 }
 return 0; // ?? (errno is set)
#else
 struct timeval tv[1];
 struct timezone tz[1];
 memset(tz, 0, sizeof(tz));
 gettimeofday(tv, tz);
 return (dword)tv->tv_sec * 1000 + tv->tv_usec / 1000;
#endif
}
#endif
```

In Core/TimeDate.cpp there is a GetSysTime, but I was missing a SetSysTime also.
Here is one that can be called from a program with administrator privileges:

```
bool SetSysTime(Time &ATime)
{
#ifdef PLATFORM_POSIX
 struct tm      tmp_time;
 tmp_time.tm_sec  = ATime.second;
 tmp_time.tm_min  = ATime.minute;
 tmp_time.tm_hour = ATime.hour;
 tmp_time.tm_mday = ATime.day;
 tmp_time.tm_mon  = ATime.month-1;
 tmp_time.tm_year = ATime.year-1900;
```

```
 time_t raw_time  = mktime(&tmp_time);

 struct timespec sys_time;
 sys_time.tv_sec  = raw_time;
 sys_time.tv_nsec = 0;

 int result = clock_settime(CLOCK_REALTIME, &sys_time);
 return (result == 0);
#endif
#ifdef PLATFORM_WIN32
 SYSTEMTIME systime;
 systime.wYear = ATime.year;
 systime.wMonth = ATime.month;
 systime.wDay = ATime.day;
 systime.wHour = ATime.hour;
 systime.wMinute = ATime.minute;
 systime.wSecond = ATime.second;
 systime.wDayOfWeek = 0;
 systime.wMilliseconds = 0;
 return SetSystemTime( &systime );
#endif
}
```

Subject: Re: Problem with GetTickCount on Linux
Posted by mirek on Wed, 30 May 2012 09:58:10 GMT
View Forum Message <> Reply to Message

Patches applied, thank you. Welcome to the contributors list.

Subject: Re: Problem with GetTickCount on Linux
Posted by Zbych on Wed, 30 May 2012 10:16:58 GMT
View Forum Message <> Reply to Message

Hi Steffen,

CLOCK_MONOTONIC is what I was looking for for
But I think it is not obligatory on all platforms.

Quote:
On POSIX systems on which these functions are available, the symbol _POSIX_TIMERS is defined in <unistd.h> to a value greater than 0. The symbols _POSIX_MONOTONIC_CLOCK, _POSIX_CPUTIME, _POSIX_THREAD_CPUTIME indicate that CLOCK_MONOTONIC, CLOCK_PROCESS_CPUTIME_ID, CLOCK_THREAD_CPUTIME_ID are available.

So maybe GetTickCount should fall back to gettimeofday in case of error?

```
#ifdef PLATFORM_POSIX
dword GetTickCount() {
#if defined(_POSIX_TIMERS) && defined(_POSIX_MONOTONIC_CLOCK)
 static bool no_monotonic = false;
 if (!no_monotonic)
 {
  struct timespec tp;
  if (clock_gettime(CLOCK_MONOTONIC, &tp) == 0)
  {
   return (dword)((tp.tv_sec * 1000) + (tp.tv_nsec / 1000000));
  }
  no_monotonic = true;
  LOG("MONOTONIC CLOCK ERROR");
 }
#endif
 struct timeval tv[1];
 struct timezone tz[1];
 memset(tz, 0, sizeof(tz));
 gettimeofday(tv, tz);
 return (dword)tv->tv_sec * 1000 + tv->tv_usec / 1000;
}

#endif
```

Subject: Re: Problem with GetTickCount on Linux
Posted by steffen on Wed, 30 May 2012 10:36:03 GMT
View Forum Message <> Reply to Message

Hi Mirek,
Thank you for the fast reponse. I see you changed the argument name in SetSysTime from ATime to time, could you please also run a search and replace to correct all the references?

Zbych,
My first version actually was calling the gettimeofday part if the call to CLOCK_MONOTONIC was failing, but I don't know if that was correct and I could not test it on any of my systems.
With the macros you found it would be safe to do something like this:

```
#ifdef PLATFORM_POSIX
```

```
dword GetTickCount() {
#if defined(_POSIX_TIMERS) && defined(_POSIX_MONOTONIC_CLOCK)
 struct timespec tp;
 if (clock_gettime(CLOCK_MONOTONIC, &tp) == 0)
 {
  return (dword)((tp.tv_sec * 1000) + (tp.tv_nsec / 1000000));
 }
#endif
 struct timeval tv[1];
 struct timezone tz[1];
 memset(tz, 0, sizeof(tz));
 gettimeofday(tv, tz);
 return (dword)tv->tv_sec * 1000 + tv->tv_usec / 1000;
}
#endif
```