

Re: MSDN volatile sample

Source: <http://www.tech-archive.net/Archive/VC/microsoft.public.vc.language/2007-12/msg01160.html>

- *From:* George <George@xxxxxxxxxxxxxxxxxxxxxxxxxxxxx>
 - *Date:* Mon, 31 Dec 2007 11:06:01 -0800
-

Thanks Ben,

Your points on Sleep is great! Cool!

regards,
George

"Ben Voigt [C++ MVP]" wrote:

"George" <George@xxxxxxxxxxxxxxxxxxxxxxxxxxxxx> wrote in message
news:4DBDB34F-8AED-4531-9AF4-581B1DD77843@xxxxxxxxxxxxxxxxxxxxx

Thanks for your clarification, Alexander!

I have got your idea. You know so many points about how thread switches internally, what materials are you referring? :-)

Just the documentation you saw. The Sleep(0) transitions a thread from "running" to "ready". No thread can ever run when a higher priority thread is "ready". Hence lower priority threads will be starved by a tight Sleep(0) loop.

A tight Sleep(50) loop is preferable in every way — it doesn't use all available CPU so lower priority threads are not starved and CPU/bus power-saving modes may operate. However, a kernel-mediated (Msg)WaitFor(Single|Multiple)Object(s)(Ex) will be more efficient any time the wait is substantial. This is why the critical section functions implement a spinloop on multiprocessors — burning some CPU cycles on a gamble that the wait will complete quickly, where the payoff is avoiding the expensive kernel-mode wait. When the wait is longer, the critical section changes to a highly-efficient kernel-mediated wait.

Re: MSDN volatile sample

regards,
George

"Alexander Grigoriev" wrote:

Sleep(0) will cause a thread dispatcher to run immediately. It will select a ready thread of highest priority and switch to it. If the only thread of highest priority ready to run is the one that called Sleep(0), it will get CPU again. Any threads of lower priority will be starved.

"George" <George@xxxxxxxxxxxxxxxxxxxxxxxxxxxx> wrote in message news:82736D9F-8E72-48EF-A126-AC324FF17524@xxxxxxxxxxxxxxxxxxxx

Hi Alexander,

Seems what you mentioned is conflicting with MSDN points. :-)

<http://msdn2.microsoft.com/en-us/library/ms686298.aspx>

If you specify 0 milliseconds, the thread will relinquish the remainder of its time slice but remain ready. Note that a ready thread is not guaranteed to run immediately.

MSDN's point is no matter the priority of threads, current thread will relinquish its remaining running time, but your points are the priority of threads matters. :-)

regards,
George

"Alexander Grigoriev" wrote:

Re: MSDN volatile sample

A thread running in a tight loop with Sleep(0) will consume all remaining CPU time, AND won't let lower priority background threads to run. It won't starve other threads of the same priority, though.

"George"

<George@xxxxxxxxxxxxxxxxxxxxxxxxxxxx>

wrote in message

news:4FB8F7B7-4452-435C-81AE-6267ED829E4C@xxxxxxxxxxxxxxxxxxxx

Thanks
Alexander,

You mean Sleep(0) will consume 100% CPU time? I can not believe it because in MSDN, it is mentioned that Sleep(0) will let current thread *If you specify 0 milliseconds, the thread will relinquish the remainder of its time slice but remain ready*. I

Re: MSDN volatile sample

assume it
means the
current
thread will
contribute
its
remaining
running
time to
other ready
status
thread.

[http://msdn2.microsoft.com/en-us/library/ms686298\(VS.85\).aspx](http://msdn2.microsoft.com/en-us/library/ms686298(VS.85).aspx)

Do you
have a test
program
which
proves
Sleep(0)
will still
occupy
100%
CPU
time?

regards,
George

"Alexander
Grigoriev"
wrote:

Sleep(0)
is
causing
kernel
trip
anyway
(lots
of
them
in
the
usual
case,
since
it's

Re: MSDN volatile sample

called
in
a
loop),
compared
to
only
2
for
WFSO/SetEvent
pair.
And
this
loop
doesn't
allow
to
stop
processor
clock
to
reduce
CPU
power.
When
I
see
an
application
consuming
100%,
that
gives
me
doubts
about
its
general
quality,
and
I
get
rid
of
it.

"Alex
Blekhman"
<tkfx.REMOVE@xxxxxxxxxx>
wrote
in

Re: MSDN volatile sample

message

[news:uFdF0\\$WSIHA.3940@xxxxxxxxxxxxxxxxxxxxxxxxxxxx](mailto:news:uFdF0$WSIHA.3940@xxxxxxxxxxxxxxxxxxxxxxxxxxxx)

"Alexander
Grigoriev"
wrote:

Anyway,
the
sample
illustrates
very
poor
synchronization
practice
which
should
not
be
followed.
Proper
signalling
using
events
makes
volatile
qualifier
unnecessary.

There
is
nothing
wrong
with
this
technique
if
it
used
correctly.
Synchronization
with
events
or
other
kernel
objects
brings
the
high

Re: MSDN volatile sample

price
of
user
space
to
kernel
space
trip.
It
is
not
uncommon
that
you
need
to
synchronize
within
single
process
only.
In
these
cases
volatile
variables
may
provide
adequate
solution.

Alex

Re: MSDN volatile sample