Министерство науки и образования РФ
Федеральное государственное автономное образовательное
учреждение высшего профессионального образования
«Санкт-Петербургский государственный электротехнический
университет «ЛЭТИ» им. В. И. Ульянова (Ленина)»
(СПбГЭТУ «ЛЭТИ»)

Кафедра вычислительной техники

# Отчёт
# по лабораторной работе № 5
# на тему:
# “Процессы и потоки в Windows”
## по дисциплине “Операционные системы”

Выполнил студент гр. 4306:
Табаков А.В.
Принял: Тимофеев А.В.

Санкт-Петербург
2016

**Цель работы**: исследовать управление файловой системой с помощью Win32 API.

**Задание 5.1.** Исследовать структуры данных процессов и потоков.

Процесс для исследования – Telegram.exe

```
0: kd> !process 0ac8 0
Searching for Process with Cid == ac8
PROCESS fffffa8008e98060
    SessionId: 1  Cid: 0ac8    Peb: 7efdf000  ParentCid: 06c0
    DirBase: 15e151000  ObjectTable: fffff8a00423e360  HandleCount: 313.
    Image: Telegram.exe
```

Список полей, составляющих блок EPROCESS:

```
0: kd> dt _eprocess
ntdll!_EPROCESS
   +0x000 Pcb              : _KPROCESS
   +0x160 ProcessLock      : _EX_PUSH_LOCK
   +0x168 CreateTime       : _LARGE_INTEGER
   +0x170 ExitTime         : _LARGE_INTEGER
   +0x178 RundownProtect   : _EX_RUNDOWN_REF
   +0x180 UniqueProcessId  : Ptr64 Void
   +0x188 ActiveProcessLinks : _LIST_ENTRY
   +0x198 ProcessQuotaUsage : [2] Uint8B
   +0x1a8 ProcessQuotaPeak : [2] Uint8B
   +0x1b8 CommitCharge     : Uint8B
   +0x1c0 QuotaBlock       : Ptr64 _EPROCESS_QUOTA_BLOCK
   +0x1c8 CpuQuotaBlock    : Ptr64 _PS_CPU_QUOTA_BLOCK
   +0x1d0 PeakVirtualSize  : Uint8B
   +0x1d8 VirtualSize      : Uint8B
   +0x1e0 SessionProcessLinks : _LIST_ENTRY
   +0x1f0 DebugPort        : Ptr64 Void
   +0x1f8 ExceptionPortData : Ptr64 Void
...
   +0x4c8 SequenceNumber    : Uint8B
   +0x4d0 CreateInterruptTime : Uint8B
   +0x4d8 CreateUnbiasedInterruptTime : Uint8B
```

EPROCESS для Telegram.exe:

```
0: kd> dt _eprocess fffffa8008e98060
ntdll!_EPROCESS
   +0x000 Pcb              : _KPROCESS
   +0x160 ProcessLock      : _EX_PUSH_LOCK
   +0x168 CreateTime       : _LARGE_INTEGER 0x01d22131`22455b5a
   +0x170 ExitTime         : _LARGE_INTEGER 0x0
   +0x178 RundownProtect   : _EX_RUNDOWN_REF
   +0x180 UniqueProcessId  : 0x00000000`00000ac8 Void
   +0x188 ActiveProcessLinks : _LIST_ENTRY [ 0xfffffa80`08eb8c98 - 0xfffffa80`08
e6d5f8 ]
   +0x198 ProcessQuotaUsage : [2] 0x7618
   +0x1a8 ProcessQuotaPeak : [2] 0x93f0
   +0x1b8 CommitCharge     : 0x3aa9
   +0x1c0 QuotaBlock       : 0xfffffa80`0838c580 _EPROCESS_QUOTA_BLOCK
...
   +0x4b0 RequestedTimerResolution : 0x8aa2bb8
   +0x4b4 ActiveThreadsHighWatermark : 0xfffffa80
   +0x4b8 SmallestTimerResolution : 0
   +0x4c0 TimerResolutionStackRecord : 0x00000000`00002710 _PO_DIAG_STACK_RECORD

   +0x4c8 SequenceNumber    : 0xfffff8a0`04d2b010
   +0x4d0 CreateInterruptTime : 0x3f
   +0x4d8 CreateUnbiasedInterruptTime : 0xd2df0a5
```

Структура KPROCESS для Telegram.exe

```
0: kd> dt _kprocess fffffa8008e98060
ntdll!_KPROCESS
   +0x000 Header           : _DISPATCHER_HEADER
   +0x018 ProfileListHead  : _LIST_ENTRY [ 0xfffffa80`08e98078 - 0xfffffa80`08e9
8078 ]
   +0x028 DirectoryTableBase : 0x00000001`5e151000
   +0x030 ThreadListHead   : _LIST_ENTRY [ 0xfffffa80`08e98e48 - 0xfffffa80`0a06
2698 ]
   +0x040 ProcessLock      : 0
   +0x048 Affinity         : _KAFFINITY_EX
   +0x070 ReadyListHead    : _LIST_ENTRY [ 0xfffffa80`08e980d0 - 0xfffffa80`08e9
80d0 ]
   +0x080 SwapListEntry    : _SINGLE_LIST_ENTRY
   +0x088 ActiveProcessors : _KAFFINITY_EX
   +0x0b0 AutoAlignment    : 0y1
   +0x0b0 DisableBoost     : 0y0
   +0x0b0 DisableQuantum   : 0y0
   +0x0b0 ActiveGroupsMask : 0y0001
   +0x0b0 ReservedFlags    : 0y00000000000000000000000000 (0)
   +0x0b0 ProcessFlags     : 0n9
   +0x0b4 BasePriority     : 8 ''
   +0x0b5 QuantumReset     : 6 ''
   +0x0b6 Visited          : 0 ''
   +0x0b7 Unused3          : 0 ''
   +0x0b8 ThreadSeed       : [4] 0
   +0x0c8 IdealNode        : [4] 0
   +0x0d0 IdealGlobalNode  : 0
   +0x0d2 Flags            : _KEXECUTE_OPTIONS
   +0x0d3 Unused1          : 0 ''
   +0x0d4 Unused2          : 0
   +0x0d8 Unused4          : 0
   +0x0dc StackCount       : _KSTACK_COUNT
   +0x0e0 ProcessListEntry : _LIST_ENTRY [ 0xfffffa80`08eb8bf0 - 0xfffffa80`08e6
d550 ]
   +0x0f0 CycleTime        : 0x615f7943
   +0x0f8 KernelTime       : 0xc
   +0x0fc UserTime         : 0x1c
   +0x100 InstrumentationCallback : (null)
   +0x108 LdtSystemDescriptor : _KGDTENTRY64
   +0x118 LdtBaseAddress   : (null)
   +0x120 LdtProcessLock   : _KGUARDED_MUTEX
   +0x158 LdtFreeSelectorHint : 0
   +0x15a LdtTableLength   : 0
```

Потоки процесса Telegram.exe:

```
0: kd> !process 0ac8 4
Searching for Process with Cid == ac8
PROCESS fffffa8008e98060
    SessionId: 1  Cid: 0ac8    Peb: 7efdf000  ParentCid: 06c0
    DirBase: 15e151000  ObjectTable: fffff8a00423e360  HandleCount: 313.
    Image: Telegram.exe

        THREAD fffffa8008e98b50  Cid 0ac8.0acc  Teb: 000000007efdb000 Win32Threa
d: fffff900c2c9dc10 WAIT
        THREAD fffffa8008efc060  Cid 0ac8.0b08  Teb: 000000007efd8000 Win32Threa
d: 0000000000000000 WAIT
        THREAD fffffa8008f9fb50  Cid 0ac8.0bb4  Teb: 000000007efd5000 Win32Threa
d: fffff900c2d9a4d0 WAIT
        THREAD fffffa8008fcea00  Cid 0ac8.0bec  Teb: 000000007efaa000 Win32Threa
d: 0000000000000000 WAIT
        THREAD fffffa8008d66060  Cid 0ac8.0c88  Teb: 000000007efa1000 Win32Threa
d: fffff900c01dec10 WAIT
        THREAD fffffa80090ff510  Cid 0ac8.0c90  Teb: 000000007ef9e000 Win32Threa
d: fffff900c2dc1c10 WAIT
        THREAD fffffa8009149b50  Cid 0ac8.0ce0  Teb: 000000007ef9b000 Win32Threa
d: fffff900c2d97c10 WAIT
        THREAD fffffa80092cb060  Cid 0ac8.0d78  Teb: 000000007ef8f000 Win32Threa
d: fffff900c2df2c10 WAIT
        THREAD fffffa80092c7060  Cid 0ac8.0d58  Teb: 000000007ef8c000 Win32Threa
d: fffff900c2dee990 WAIT
        THREAD fffffa80061fbb50  Cid 0ac8.0820  Teb: 000000007ef92000 Win32Threa
d: fffff900c0607010 WAIT
        THREAD fffffa80061fe3d0  Cid 0ac8.0fbc  Teb: 000000007ef89000 Win32Threa
d: fffff900c070d980 WAIT
        THREAD fffffa8006200390  Cid 0ac8.0fcc  Teb: 000000007ef86000 Win32Threa
d: 0000000000000000 WAIT
        THREAD fffffa8006b81b50  Cid 0ac8.0fd4  Teb: 000000007ef83000 Win32Threa
d: fffff900c32124e0 WAIT
        THREAD fffffa8006d20b50  Cid 0ac8.0ce4  Teb: 000000007ef80000 Win32Threa
d: fffff900c2df2320 WAIT
        THREAD fffffa8006ce3b50  Cid 0ac8.0e9c  Teb: 000000007ef7a000 Win32Threa
d: fffff900c2de8c10 WAIT
        THREAD fffffa8006d42060  Cid 0ac8.0d28  Teb: 000000007ef77000 Win32Threa
d: fffff900c2df2810 WAIT
        THREAD fffffa800a2d9b50  Cid 0b2c.0ecc  Teb: 0000000000000000 Win32Threa
d: 0000000000000000 TERMINATED
TYPE mismatch for thread object at fffffa8008f3b848
```

Подробная информация о первом потоке:

```
0: kd> !thread fffffa8008e98b50
THREAD fffffa8008e98b50  Cid 0ac8.0acc  Teb: 000000007efdb000 Win32Thread: fffff
900c2c9dc10 WAIT: (UserRequest) UserMode Alertable
    fffffa8008fa7940  NotificationEvent
    fffffa8008e9f360  SynchronizationEvent
IRP List:
    fffffa8008f76c60: (0006,0118) Flags: 00060800  Mdl: 00000000
    fffffa8008f76ee0: (0006,0118) Flags: 00060800  Mdl: 00000000
    fffffa8008f64010: (0006,0118) Flags: 00060800  Mdl: 00000000
    fffffa8008fa7d90: (0006,0118) Flags: 00060800  Mdl: 00000000
    fffffa80084ec1a0: (0006,0118) Flags: 00060800  Mdl: 00000000
    fffffa8008369b00: (0006,0118) Flags: 00060800  Mdl: 00000000
    fffffa8008d5f410: (0006,0118) Flags: 00060800  Mdl: 00000000
    fffffa8008dcf140: (0006,0118) Flags: 00060800  Mdl: 00000000
Not impersonating
DeviceMap                 fffff8a0016b0a30
Owning Process            fffffa8008e98060       Image:         Telegram.exe
Attached Process          N/A            Image:         N/A
Wait Start TickCount      337748
Context Switch Count      78155          IdealProcessor: 2                 Large
Stack
UserTime                  00:00:03.198
KernelTime                00:00:02.605
Win32 Start Address 0x0000000000edf06c
Stack Init fffff8800918bc70 Current fffff8800918ae80
Base fffff8800918c000 Limit fffff88009182000 Call 0
Priority 10 BasePriority 8 UnusualBoost 0 ForegroundBoost 2 IoPriority 2 PagePri
ority 5
Child-SP          RetAddr           : Args to Child
                                    : Call Site
fffff880`0918aec0 fffff800`02e69db2 : 00000000`00000202 fffffa80`08e98b50 fffff8
00`00000000 fffffa80`08db4b50 : nt!KiSwapContext+0x7a
fffff880`0918b000 fffff800`02e768da : fffffa80`08326610 fffff800`02e5400c fffff8
80`00000000 fffffa80`08db4c58 : nt!KiCommitThreadWait+0x1d2
fffff880`0918b090 fffff800`0316cdff : fffff880`00000002 fffff880`0918b3e0 000000
00`00000001 fffff800`00000006 : nt!KeWaitForMultipleObjects+0x272
fffff880`0918b350 fffff800`0319ab29 : 00000000`00000001 fffff800`033f2b7f 000000
00`00000001 fffff960`0013f401 : nt!ObpWaitForMultipleObjects+0x294
fffff880`0918b820 fffff800`02e73613 : fffff880`0918bb60 fffff800`02e7d5ce 000000
00`0000002a 00000000`72db2450 : nt!NtWaitForMultipleObjects32+0xec
fffff880`0918ba70 00000000`72db2e09 : 00000000`00000000 00000000`00000000 000000
00`00000000 00000000`00000000 : nt!KiSystemServiceCopyEnd+0x13 (TrapFrame @ ffff
f880`0918bae0)
00000000`0244e508 00000000`00000000 : 00000000`00000000 00000000`00000000 000000
00`00000000 00000000`00000000 : 0x72db2e09
```

Значения полей ETHREAD первого потока Telegram.exe

```
0: kd> dt _ethread fffffa8008e98b50
ntdll!_ETHREAD
   +0x000 Tcb              : _KTHREAD
   +0x368 CreateTime       : _LARGE_INTEGER 0x01d22131`22455b5a
   +0x370 ExitTime         : _LARGE_INTEGER 0xfffffa80`08e98ec0
   +0x370 KeyedWaitChain   : _LIST_ENTRY [ 0xfffffa80`08e98ec0 - 0xfffffa80`08e9
8ec0 ]
   +0x380 ExitStatus       : 0n0
   +0x388 PostBlockList    : _LIST_ENTRY [ 0x00000000`00000000 - 0x00000000`76cf
a2c0 ]
   +0x388 ForwardLinkShadow : (null)
   +0x390 StartAddress     : 0x00000000`76cfa2c0 Void
   +0x398 TerminationPort  : (null)
   +0x398 ReaperLink       : (null)
   +0x398 KeyedWaitValue   : (null)
   +0x3a0 ActiveTimerListLock : 0
   +0x3a8 ActiveTimerListHead : _LIST_ENTRY [ 0xfffffa80`08e98ef8 - 0xfffffa80`0
8e98ef8 ]
   +0x3b8 Cid              : _CLIENT_ID
   +0x3c8 KeyedWaitSemaphore : _KSEMAPHORE
   +0x3c8 AlpcWaitSemaphore : _KSEMAPHORE
   +0x3e8 ClientSecurity   : _PS_CLIENT_SECURITY_CONTEXT
   +0x3f0 IrpList          : _LIST_ENTRY [ 0xfffffa80`08f76c80 - 0xfffffa80`08dc
f160 ]
...
   +0x498 CmCallbackListHead : _SINGLE_LIST_ENTRY
   +0x4a0 KernelStackReference : 1
```

**Вывод**: при помощи livekd можно просматривать список EPROCESS, получать список потоков процесса и просматривать информацию о них.

**Задание 5.2.** Исследовать регистр контроля процессора и очередь потоков готовых для выполнения.

Регистр контроля ядер процессора:

```
0: kd> !prcb
PRCB for Processor 0 at fffff80002ff2e80:
Current IRQL -- 0
Threads--  Current fffffa800834bb50 Next 0000000000000000 Idle fffff80003000cc0
Processor Index 0 Number (0, 0) GroupSetMember 1
Interrupt Count -- 004465bc
Times -- Dpc     00000306 Interrupt 00000442
         Kernel 00049cfe User      00007401
```

…

```
0: kd> !prcb 3
PRCB for Processor 3 at fffff88002fd7180:
Current IRQL -- 0
Threads--  Current fffff88002fe1fc0 Next 0000000000000000 Idle fffff88002fe1fc0
Processor Index 3 Number (0, 3) GroupSetMember 8
Interrupt Count -- 00582b46
Times -- Dpc     000005d9 Interrupt 00000e56
         Kernel 0005d6a5 User      0000b151
```

Kernel Processor Register Control Block для ядра 0:

```
0: kd> dt _kprcb fffff80002ff2e80
ntdll!_KPRCB
   +0x000 MxCsr            : 0x1f80
   +0x004 LegacyNumber     : 0 ''
   +0x005 ReservedMustBeZero : 0 ''
   +0x006 InterruptRequest : 0 ''
   +0x007 IdleHalt         : 0 ''
   +0x008 CurrentThread    : 0xfffffa80`0a58fb50 _KTHREAD
   +0x010 NextThread       : (null)
   +0x018 IdleThread       : 0xfffff800`03000cc0 _KTHREAD
   +0x020 NestingLevel     : 0 ''
   +0x021 PrcbPad00        : [3]  ""
   +0x024 Number           : 0
   +0x028 RspBase          : 0xfffff880`08eaec70
   +0x030 PrcbLock         : 0
   +0x038 PrcbPad01        : 0
   +0x040 ProcessorState   : _KPROCESSOR_STATE
   +0x5f0 CpuType          : 21 ''
   +0x5f1 CpuID            : 1 ''
   +0x5f2 CpuStep          : 0x1301
   +0x5f2 CpuStepping      : 0x1 ''
```

…

```
+0x4be8 ExtendedState    : 0xfffff880`009e4000 _XSAVE_AREA
+0x4c00 Mailbox          : (null)
+0x4c80 RequestMailbox   : [1] _REQUEST_MAILBOX
```

У данного ядра:

```
+0x4498 ReadySummary: 0x100

   +0x4500 DispatcherReadyListHead : [32] _LIST_ENTRY [ 0xfffff800`02ff7380 - 0x
fffff800`02ff7380 ]
```

DispatcherReadtListHead в памяти:

```
0: kd> dd fffff80002ff2e80+4500
fffff800`02ff7380  02ff7380 fffff800 02ff7380 fffff800
fffff800`02ff7390  02ff7390 fffff800 02ff7390 fffff800
fffff800`02ff73a0  02ff73a0 fffff800 02ff73a0 fffff800
fffff800`02ff73b0  02ff73b0 fffff800 02ff73b0 fffff800
fffff800`02ff73c0  02ff73c0 fffff800 02ff73c0 fffff800
fffff800`02ff73d0  02ff73d0 fffff800 02ff73d0 fffff800
fffff800`02ff73e0  02ff73e0 fffff800 02ff73e0 fffff800
fffff800`02ff73f0  02ff73f0 fffff800 02ff73f0 fffff800
```

Рекурсивный просмотр содержимого:

```
0: kd> dt _list_entry fffff800`02ff7380 -r1
ntdll!_LIST_ENTRY
 [ 0xfffff800`02ff7380 - 0xfffff800`02ff7380 ]
   +0x000 Flink            : 0xfffff800`02ff7380 _LIST_ENTRY [ 0xfffff800`02ff73
80 - 0xfffff800`02ff7380 ]
      +0x000 Flink          : 0xfffff800`02ff7380 _LIST_ENTRY [ 0xfffff800`02f
f7380 - 0xfffff800`02ff7380 ]
      +0x008 Blink          : 0xfffff800`02ff7380 _LIST_ENTRY [ 0xfffff800`02f
f7380 - 0xfffff800`02ff7380 ]
   +0x008 Blink            : 0xfffff800`02ff7380 _LIST_ENTRY [ 0xfffff800`02ff73
80 - 0xfffff800`02ff7380 ]
      +0x000 Flink          : 0xfffff800`02ff7380 _LIST_ENTRY [ 0xfffff800`02f
f7380 - 0xfffff800`02ff7380 ]
      +0x008 Blink          : 0xfffff800`02ff7380 _LIST_ENTRY [ 0xfffff800`02f
f7380 - 0xfffff800`02ff7380 ]
```

KTHREAD для 0 ядра:

```
0: kd> dt _kthread 0xfffff800`02ff7380+0xFFFFFFFFFFFFF8C
ntdll!_KTHREAD
   +0x000 Header           : _DISPATCHER_HEADER
   +0x018 CycleTime        : 0x02ff5000`00000000
   +0x020 QuantumTarget    : 0x02ff5000`fffff800
   +0x028 InitialStack     : 0x02fa3160`fffff800 Void
   +0x030 StackLimit       : 0x00000000`fffff800 Void
   +0x038 KernelStack      : 0xffffff6c`00000000 Void
   +0x040 ThreadLock       : 0
   +0x048 WaitRegister     : _KWAIT_STATUS_REGISTER
   +0x049 Running          : 0 ''
   +0x04a Alerted          : [2]  ""
   +0x04c KernelStackResident : 0y0
   +0x04c ReadyTransition  : 0y0
   +0x04c ProcessReadyQueue : 0y0
   +0x04c WaitNext         : 0y0
   +0x04c SystemAffinityActive : 0y0
   +0x04c Alertable        : 0y0
   +0x04c GdiFlushActive   : 0y0
   +0x04c UserStackWalkActive : 0y0

...

+0x059 Priority       :8 ''


...

+0x210 Process          : 0x02ff7520`fffff800 _KPROCESS

...

+0x350 ThreadCounters   : (null)
+0x358 StateSaveArea    : 0x00000000`0000265f _XSAVE_FORMAT
+0x360 XStateSave       : (null)
```

EPROCESS

```
0: kd> dt _eprocess fffff80002ff7520
ntdll!_EPROCESS
   +0x000 Pcb              : _KPROCESS
   +0x160 ProcessLock      : _EX_PUSH_LOCK
   +0x168 CreateTime       : _LARGE_INTEGER 0x12f7
   +0x170 ExitTime         : _LARGE_INTEGER 0x00000003`18bdc77f
   +0x178 RundownProtect   : _EX_RUNDOWN_REF
   +0x180 UniqueProcessId  : 0xffffa80`06ed6b10 Void
   +0x188 ActiveProcessLinks : _LIST_ENTRY [ 0x00000000`00000000 - 0x0005d73f`00
000164 ]
   +0x198 ProcessQuotaUsage : [2] 0xfffff800`0320bc80
   +0x1a8 ProcessQuotaPeak : [2] 1
   +0x1b8 CommitCharge     : 0
   +0x1c0 QuotaBlock       : (null)
   +0x1c8 CpuQuotaBlock    : 0x00000000`01000313 _PS_CPU_QUOTA_BLOCK
   +0x1d0 PeakVirtualSize  : 0xfffff800`02ff5000
   +0x1d8 VirtualSize      : 0xfffff800`02ff5000
```

…

```
   +0x4a0 TimerResolutionLink : _LIST_ENTRY [ 0x002f6374`003d8198 - 0x04ea17bc`0
0000231 ]
   +0x4b0 RequestedTimerResolution : 0x90e5
   +0x4b4 ActiveThreadsHighWatermark : 0x4e406db
   +0x4b8 SmallestTimerResolution : 0x57fe3
   +0x4c0 TimerResolutionStackRecord : 0x0002212b`00000008 _PO_DIAG_STACK_RECORD

   +0x4c8 SequenceNumber    : 0x000200e4`0000000b
   +0x4d0 CreateInterruptTime : 0x00000003`0000203c
   +0x4d8 CreateUnbiasedInterruptTime : 0
```

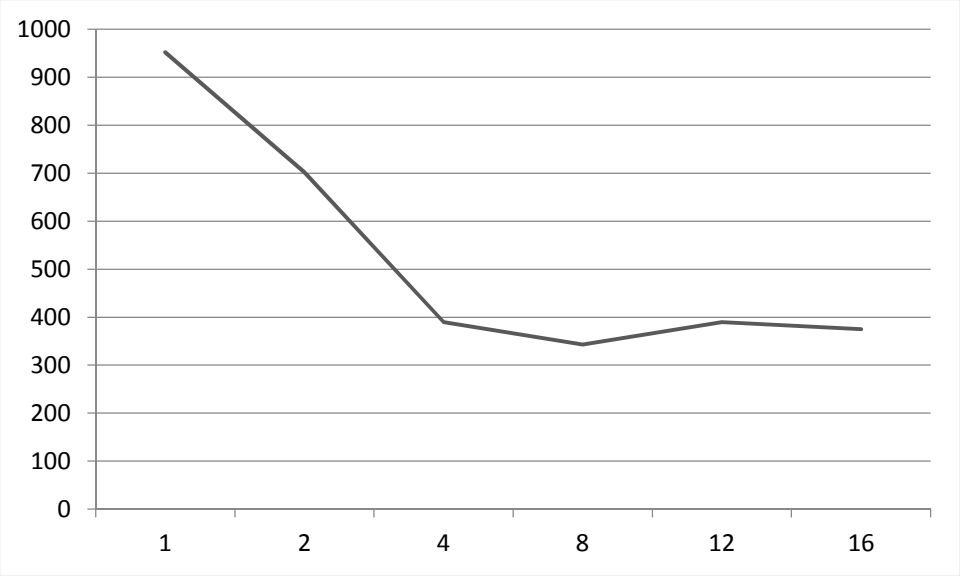
**Вывод**: при помощи livekd можно узнать какие приоритеты у потоков на выполнение и каким процессам они принадлежат.

**Задание 5.3.** Реализация многопоточного приложения с использованием функций Win32 API.

$$\pi = \left( \frac{4}{1+x_0^2} + \frac{4}{1+x_1^2} + \ldots + \frac{4}{1+x_{N-1}^2} \right) \times \frac{1}{N}, \text{ где } x_i = (i+0.5) \times \frac{1}{N}, \; i = \overline{0, N-1}$$

**где N=100000000**

И размер блока 10*500



```
0: kd> !process 1fd8
*** ERROR: Module load completed but symbols could not be loaded for LiveKdD.SYS

Searching for Process with Cid == 1fd8
PROCESS fffffa800891f410
    SessionId: 1  Cid: 1fd8    Peb: 7ffffdf000  ParentCid: 2980
    DirBase: 3102b000  ObjectTable: fffff8a0187c5ae0  HandleCount:  25.
    Image: 5. Au?eneaiea Ie.exe
    VadRoot fffffa80064c8010 Vads 61 Clone 0 Private 298. Modified 0. Locked 0.
    DeviceMap fffff8a002e9b680
    Token                             fffff8a0197b2060
    ElapsedTime                       00:00:14.504
    UserTime                          00:00:00.000
    KernelTime                        00:00:00.000
    QuotaPoolUsage[PagedPool]         84144
    QuotaPoolUsage[NonPagedPool]      7200
    Working Set Sizes (now,min,max)  (1146, 50, 345) (4584KB, 200KB, 1380KB)
    PeakWorkingSetSize                1146
    VirtualSize                       48 Mb
    PeakVirtualSize                   48 Mb
    PageFaultCount                    1161
    MemoryPriority                    BACKGROUND
    BasePriority                      8
    CommitCharge                      372
    DebugPort                         fffffa800abaab70

        THREAD fffffa8006b3eb50  Cid 1fd8.1bcc  Teb: 000007fffffdd000 Win32Threa
d: fffff900c2e77c10 WAIT: (Executive) KernelMode Non-Alertable
```

```
0: kd> dt _eprocess fffffa800891f410
nt!_EPROCESS
   +0x000 Pcb              : _KPROCESS
   +0x160 ProcessLock      : _EX_PUSH_LOCK
   +0x168 CreateTime       : _LARGE_INTEGER 0x01d2424f`61398abc
   +0x170 ExitTime         : _LARGE_INTEGER 0x0
   +0x178 RundownProtect   : _EX_RUNDOWN_REF
   +0x180 UniqueProcessId  : 0x00000000`00001fd8 Void
   +0x188 ActiveProcessLinks : _LIST_ENTRY [ 0xfffffa80`09eac1e8 - 0xfffffa80`0a
9be618 ]
   +0x198 ProcessQuotaUsage : [2] 0x1c20
   +0x1a8 ProcessQuotaPeak : [2] 0x1c20
   +0x1b8 CommitCharge     : 0x174
   +0x1c0 QuotaBlock       : 0xfffffa80`083afd80 _EPROCESS_QUOTA_BLOCK
   +0x1c8 CpuQuotaBlock    : (null)
   +0x1d0 PeakVirtualSize  : 0x3078000
   +0x1d8 VirtualSize      : 0x3078000
   +0x1e0 SessionProcessLinks : _LIST_ENTRY [ 0xfffffa80`09eac240 - 0xfffffa80`0
a9be670 ]
   +0x1f0 DebugPort        : 0xfffffa80`0abaab70 Void
   +0x1f8 ExceptionPortData : 0xfffffa80`0831ee60 Void
   +0x1f8 ExceptionPortValue : 0xfffffa80`0831ee60
   +0x1f8 ExceptionPortState : 0y000
   +0x200 ObjectTable      : 0xfffff8a0`187c5ae0 _HANDLE_TABLE
   +0x208 Token            : _EX_FAST_REF
```

...

```
 +0x4d8 CreateInterruptTime : 0x00000044`cb05f0ee
 +0x4e0 CreateUnbiasedInterruptTime : 0x0000003b`8deb2fb5
```

```
0: kd> dt _kprocess fffffa800891f410
nt!_KPROCESS
   +0x000 Header           : _DISPATCHER_HEADER
   +0x018 ProfileListHead  : _LIST_ENTRY [ 0xfffffa80`0891f428 - 0xfffffa80`0891
f428 ]
   +0x028 DirectoryTableBase : 0x3102b000
   +0x030 ThreadListHead   : _LIST_ENTRY [ 0xfffffa80`06b3ee48 - 0xfffffa80`0a60
1c68 ]
   +0x040 ProcessLock      : 0
   +0x048 Affinity         : _KAFFINITY_EX
   +0x070 ReadyListHead    : _LIST_ENTRY [ 0xfffffa80`0891f480 - 0xfffffa80`0891
f480 ]
   +0x080 SwapListEntry    : _SINGLE_LIST_ENTRY
   +0x088 ActiveProcessors : _KAFFINITY_EX
   +0x0b0 AutoAlignment    : 0y0
   +0x0b0 DisableBoost     : 0y0
```

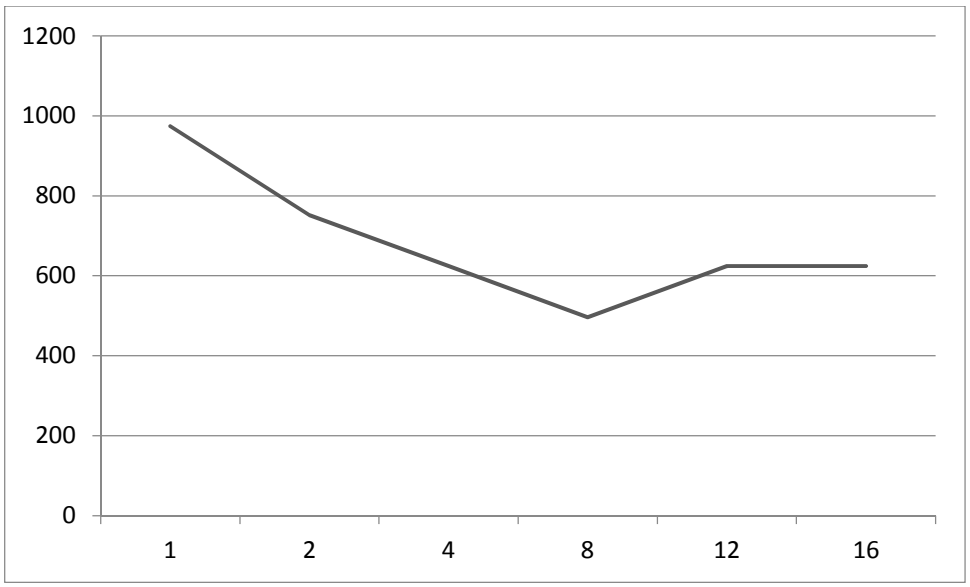
```
0: kd> !thread fffffa8006af5060
THREAD fffffa8006af5060  Cid 1fd8.2884  Teb: 000007fffffac000 Win32Thread: 00000
00000000000 WAIT: (Suspended) KernelMode Non-Alertable
FreezeCount 1
    fffffa8006af5338  Semaphore Limit 0x2
Not impersonating
DeviceMap                 fffff8a002e9b680
Owning Process            fffffa800891f410       Image:         5. Au?eneaiea Ie
.exe
Attached Process          N/A            Image:         N/A
Wait Start TickCount      1894228        Ticks: 689 (0:00:00:10.748)
Context Switch Count      14             IdealProcessor: 0
UserTime                  00:00:00.000
KernelTime                00:00:00.000
Win32 Start Address 0x000000013f261280
Stack Init fffff8800c92ec70 Current fffff8800c92e5d0
Base fffff8800c92f000 Limit fffff8800c929000 Call 0
Priority 10 BasePriority 8 UnusualBoost 0 ForegroundBoost 2 IoPriority 2 PagePri
ority 5
Child-SP          RetAddr           : Args to Child
                                    : Call Site
fffff880`0c92e610 fffff800`02e83db2 : fffffa80`06af5120 fffffa80`06af5060 000000
00`00000000 fffffa80`06af5060 : nt!KiSwapContext+0x7a
fffff880`0c92e750 fffff800`02e951cf : 00000000`00000000 fffffa80`06af5000 fffffa
80`00000000 fffff800`02e9127a : nt!KiCommitThreadWait+0x1d2
fffff880`0c92e7e0 fffff800`02e80804 : 00000000`00000000 00000000`00000005 000000
00`00000000 00000000`00000000 : nt!KeWaitForSingleObject+0x19f
fffff880`0c92e880 fffff800`02e814ad : fffffa80`06af5060 00000000`00000000 fffff8
00`0300ce80 00000000`00000000 : nt!KiSuspendThread+0x54
fffff880`0c92e8c0 fffff800`02e83fbd : fffffa80`06af5120 00000000`00000000 fffff8
00`02e807b0 00000000`00000000 : nt!KiDeliverApc+0x21d
fffff880`0c92e940 fffff800`02e951cf : 00000000`00000064 fffffa80`0891f410 000000
00`00000000 fffff8a0`0b6391a0 : nt!KiCommitThreadWait+0x3dd
fffff880`0c92e9d0 fffff800`03186ace : fffffa80`0891f400 00000000`00000006 000000
00`00000001 fffff800`03187c00 : nt!KeWaitForSingleObject+0x19f
fffff880`0c92ea70 fffff800`02e8d613 : fffffa80`06af5060 00000000`00000064 000000
00`00000000 fffffa80`083968d0 : nt!NtWaitForSingleObject+0xde
fffff880`0c92eae0 00000000`77aabb7a : 00000000`00000000 00000000`00000000 000000
00`00000000 00000000`00000000 : nt!KiSystemServiceCopyEnd+0x13 (TrapFrame @ ffff
f880`0c92eae0)
00000000`02b5f0a8 00000000`00000000 : 00000000`00000000 00000000`00000000 000000
00`00000000 00000000`00000000 : 0x77aabb7a
```

**Вывод:** WINAPI позволяет создавать многопоточные приложения, но для их разработки требуется большая внимательность, потому что очень легко ошибиться и всё будет работать неправильно.

**Задание 5.4.** Реализация многопоточного приложения с использованием технологии OpenMP.

$$\pi = \left( \frac{4}{1+x_0^2} + \frac{4}{1+x_1^2} + \ldots + \frac{4}{1+x_{N-1}^2} \right) \times \frac{1}{N}, \text{ где } x_i = (i+0.5) \times \frac{1}{N}, \; i = \overline{0, N-1}$$

где N=10000000.




```
0: kd> !process 498
Searching for Process with Cid == 498
PROCESS fffffa800adb4370
    SessionId: 1  Cid: 0498    Peb: 7fffffdf000  ParentCid: 574c
    DirBase: 13782000  ObjectTable: fffff8a009f00980  HandleCount:  18.
    Image: 5. Au?eneaiea Ie.exe
    VadRoot fffffa800656dc40 Vads 45 Clone 0 Private 266. Modified 0. Locked 0.
    DeviceMap fffff8a0016b0a30
    Token                             fffff8a01a9a19d0
    ElapsedTime                       <Invalid>
    UserTime                          00:00:00.000
    KernelTime                        00:00:00.000
    QuotaPoolUsage[PagedPool]         84144
    QuotaPoolUsage[NonPagedPool]      5280
    Working Set Sizes (now,min,max)  (1124, 50, 345) (4496KB, 200KB, 1380KB)
    PeakWorkingSetSize                1124
    VirtualSize                       40 Mb
    PeakVirtualSize                   40 Mb
    PageFaultCount                    1139
    MemoryPriority                    BACKGROUND
    BasePriority                      8
    CommitCharge                      317
    DebugPort                         fffffa8009788830

        THREAD fffffa800b510b50  Cid 0498.563c  Teb: 000007fffffdd000 Win32Threa
d: fffff900c31a3c10 WAIT: (UserRequest) UserMode Non-Alertable
            fffffa800b09c060  ProcessObject
        Not impersonating
        DeviceMap                 fffff8a0016b0a30
        Owning Process            fffffa800adb4370       Image:         5. Au?en
eaiea Ie.exe
        Attached Process          N/A            Image:         N/A
        Wait Start TickCount      1434972
        Context Switch Count      596            IdealProcessor: 2
   LargeStack
        UserTime                  00:00:00.078
        KernelTime                00:00:00.015
        Win32 Start Address 0x000000013f14117c
        Stack Init fffff8800b75cc70 Current fffff8800b75c7c0
        Base fffff8800b75d000 Limit fffff8800b755000 Call 0
        Priority 11 BasePriority 8 UnusualBoost 0 ForegroundBoost 2 IoPriority 2
 PagePriority 5
        Child-SP          RetAddr           Call Site
        fffff880`0b75c800 fffff800`02e69db2 nt!KiSwapContext+0x7a
        fffff880`0b75c940 fffff800`02e7b1cf nt!KiCommitThreadWait+0x1d2
        fffff880`0b75c9d0 fffff800`0316cace nt!KeWaitForSingleObject+0x19f
        fffff880`0b75ca70 fffff800`02e73613 nt!NtWaitForSingleObject+0xde
        fffff880`0b75cae0 00000000`76d1bb7a nt!KiSystemServiceCopyEnd+0x13 (Trap
Frame @ fffff880`0b75cae0)
        00000000`0025f338 00000000`00000000 0x76d1bb7a
```

```
0: kd> dt _eprocess  fffffa800adb4370
nt!_EPROCESS
   +0x000 Pcb              : _KPROCESS
   +0x160 ProcessLock      : _EX_PUSH_LOCK
   +0x168 CreateTime       : _LARGE_INTEGER 0x01d22165`32618fb1
   +0x170 ExitTime         : _LARGE_INTEGER 0x0
   +0x178 RundownProtect   : _EX_RUNDOWN_REF
   +0x180 UniqueProcessId  : 0x00000000`00000498 Void
   +0x188 ActiveProcessLinks : _LIST_ENTRY [ 0xfffffa80`097671e8 - 0xfffffa80`0b
0907b8 ]
   +0x198 ProcessQuotaUsage : [2] 0x14a0
   +0x1a8 ProcessQuotaPeak : [2] 0x1520
   +0x1b8 CommitCharge     : 0x13d
   +0x1c0 QuotaBlock       : 0xfffffa80`0838c580 _EPROCESS_QUOTA_BLOCK
   +0x1c8 CpuQuotaBlock    : (null)
   +0x1d0 PeakVirtualSize  : 0x2878000
   +0x1d8 VirtualSize      : 0x2878000
   +0x1e0 SessionProcessLinks : _LIST_ENTRY [ 0xfffffa80`09767240 - 0xfffffa80`0
```

...

```
+0x4d8 CreateInterruptTime : 0x00000034`1d4a24fc
+0x4e0 CreateUnbiasedInterruptTime : 0x00000034`1ced10e3
```

```
0: kd> dt _kprocess  fffffa800adb4370
nt!_KPROCESS
   +0x000 Header           : _DISPATCHER_HEADER
   +0x018 ProfileListHead  : _LIST_ENTRY [ 0xfffffa80`0adb4388 - 0xfffffa80`0adb
4388 ]
   +0x028 DirectoryTableBase : 0x13782000
   +0x030 ThreadListHead   : _LIST_ENTRY [ 0xfffffa80`0b510e48 - 0xfffffa80`0b51
0e48 ]
   +0x040 ProcessLock      : 0
   +0x048 Affinity         : _KAFFINITY_EX
   +0x070 ReadyListHead    : _LIST_ENTRY [ 0xfffffa80`0adb43e0 - 0xfffffa80`0adb
43e0 ]
   +0x080 SwapListEntry    : _SINGLE_LIST_ENTRY
   +0x088 ActiveProcessors : _KAFFINITY_EX
   +0x0b0 AutoAlignment    : 0y0
   +0x0b0 DisableBoost     : 0y0
   +0x0b0 DisableQuantum   : 0y0
   +0x0b0 ActiveGroupsMask : 0y0001
   +0x0b0 ReservedFlags    : 0y000000000000000000000000 (0)
   +0x0b0 ProcessFlags     : 0n8
   +0x0b4 BasePriority     : 8 ''
   +0x0b5 QuantumReset     : 6 ''
   +0x0b6 Visited          : 0 ''
   +0x0b7 Unused3          : 0 ''
   +0x0b8 ThreadSeed       : [4] 2
   +0x0c8 IdealNode        : [4] 0
   +0x0d0 IdealGlobalNode  : 0
   +0x0d2 Flags            : _KEXECUTE_OPTIONS
   +0x0d3 Unused1          : 0 ''
   +0x0d4 Unused2          : 0
   +0x0d8 Unused4          : 0
   +0x0dc StackCount       : _KSTACK_COUNT
   +0x0e0 ProcessListEntry : _LIST_ENTRY [ 0xfffffa80`09767140 - 0xfffffa80`0b09
0710 ]
   +0x0f0 CycleTime        : 0
   +0x0f8 KernelTime       : 0
   +0x0fc UserTime         : 0
   +0x100 InstrumentationCallback : (null)
   +0x108 LdtSystemDescriptor : _KGDTENTRY64
   +0x118 LdtBaseAddress   : (null)
   +0x120 LdtProcessLock   : _KGUARDED_MUTEX
   +0x158 LdtFreeSelectorHint : 0
   +0x15a LdtTableLength   : 0
```

```
0: kd> !thread fffffa800b510b50
*** ERROR: Module load completed but symbols could not be loaded for LiveKdD.SYS

THREAD fffffa800b510b50  Cid 0498.563c  Teb: 000007fffffdd000 Win32Thread: fffff
900c31a3c10 WAIT: (UserRequest) UserMode Non-Alertable
    fffffa800b09c060  ProcessObject
Not impersonating
DeviceMap                 fffff8a0016b0a30
Owning Process            fffffa800adb4370       Image:         5. Au?eneaiea Ie
.exe
Attached Process          N/A            Image:         N/A
Wait Start TickCount      1434972        Ticks: 23796 (0:00:06:11.219)
Context Switch Count      596            IdealProcessor: 2                 Large
Stack
UserTime                  00:00:00.078
KernelTime                00:00:00.015
Win32 Start Address 0x000000013f14117c
Stack Init fffff8800b75cc70 Current fffff8800b75c7c0
Base fffff8800b75d000 Limit fffff8800b755000 Call 0
Priority 11 BasePriority 8 UnusualBoost 0 ForegroundBoost 2 IoPriority 2 PagePri
ority 5
Kernel stack not resident.
Child-SP          RetAddr           : Args to Child
                              : Call Site
fffff880`0b75c800 fffff800`02e69db2 : fffffa80`0aa44b40 fffffa80`0b510b50 fffffa
80`00000000 fffff880`0b75c958 : nt!KiSwapContext+0x7a
fffff880`0b75c940 fffff800`02e7b1cf : 00000000`00000040 fffffa80`0adb4370 000000
00`00000000 fffff8a0`000ed120 : nt!KiCommitThreadWait+0x1d2
fffff880`0b75c9d0 fffff800`0316cace : fffffa80`0adb4300 00000000`00000006 000000
00`00000001 fffff800`0316dc00 : nt!KeWaitForSingleObject+0x19f
fffff880`0b75ca70 fffff800`02e73613 : fffffa80`0b510b50 00000000`ffffffff 000000
00`00000000 fffffa80`0b09c060 : nt!NtWaitForSingleObject+0xde
fffff880`0b75cae0 00000000`76d1bb7a : 00000000`00000000 00000000`00000000 000000
00`00000000 00000000`00000000 : nt!KiSystemServiceCopyEnd+0x13 (TrapFrame @ ffff
f880`0b75cae0)
00000000`0025f338 00000000`00000000 : 00000000`00000000 00000000`00000000 000000
00`00000000 00000000`00000000 : 0x76d1bb7a
```

**Вывод:** Библиотека OpenMP позволяет значительно упростить жизнь разработчикам, за счёт своей простоты.

## Приложение

## Текст программы

```
#include <windows.h>
#include <iostream>
#include <iomanip>
#include <omp.h>
#include <AclAPI.h>
#include <AccCtrl.h>

using namespace std;

typedef struct _Thread {
	HANDLE hThread;
	volatile unsigned int nextBlockIndex;
	double threadPi;
	volatile bool finished;
	volatile bool calculating;
} Thread;

CRITICAL_SECTION crit;
const int BLOCK_SIZE = 10*500;
const int N = 100000000;
const int TOTAL_BLOCKS = N / BLOCK_SIZE+(N % BLOCK_SIZE ? 1 : 0);

Thread *pThreads;


int menu();
void win32Processing();
DWORD WINAPI worker(LPVOID);
double OMP();
__declspec(thread) DWORD dwTlsThreadIndex;

int main()
{

	setlocale(0, ".1251");
	int notExit;

	switch (notExit = menu())
	{
	case 1:
		win32Processing();
		break;
	case 2:
		OMP();
		break;
	case 0:
		break;
	default:
		if (notExit)
			cout << "Такого варианта нет" << endl;
	}

	system("pause");
	return 0;
}
```

```
int menu()
{
	system("cls");
	int point;
	do {
		cin.clear();
		cin.sync();

		cout << "Выберите пункт меню" << endl;
		cout << "1 - Win32 API" << endl;
		cout << "2 - Open Multi-Processing" << endl;
		cout << "0 - Выход" << endl;
		cout << ">";
		cin >> point;
		if (cin.fail())
			cout << "Что-то пошло не так, выберите пункт меню повторно" << endl;
	} while (cin.fail());
	system("cls");
	return point;

}
volatile LONG nextBlock = 0;
int numOfThreads = 1;
//int* iterationsPerThread;
void win32Processing() {
	srand(time(NULL));
	double pi = 0, start = 0, end = 0;
	cout << "Всего блоков: " << TOTAL_BLOCKS << "\n";
	cout << "Потоков : ";
	cin >> numOfThreads;
	//iterationsPerThread = new int[numOfThreads];
	pThreads = new Thread[numOfThreads];
	for (int i = 0; i < numOfThreads; ++i) {
		//iterationsPerThread [i] = 0;
		pThreads[i].nextBlockIndex = nextBlock++;
		pThreads[i].threadPi = 0;
		pThreads[i].finished = false;
		pThreads[i].calculating = false;
		pThreads[i].hThread = CreateThread(NULL, 0, worker, (LPVOID)i, CREATE_SUSPENDED,
NULL);
	}
	HANDLE* handlesArray = new HANDLE[numOfThreads];

	for (int i = 0; i < numOfThreads; ++i) {
		handlesArray[i] = pThreads[i].hThread;
	}

	start = GetTickCount();
	for (int i = 0; i < numOfThreads; i++) {
		ResumeThread(pThreads[i].hThread);
	}

	while (nextBlock <= TOTAL_BLOCKS) {
		int i;
		//rand() % numOfThreads
		for (i = 0; ; i = (i + 1) % numOfThreads) {
			SwitchToThread();
```

```
                        if (!pThreads[i].calculating || nextBlock > TOTAL_BLOCKS)
                                break;
                }
                ResumeThread(pThreads[i].hThread);
        }
        cout << nextBlock << endl;
        for (int i = 0; i < numOfThreads; i++){
                ResumeThread(pThreads[i].hThread);
                cout << i << "  " << a[i] << endl;
        }
        DWORD check = WaitForMultipleObjects(numOfThreads, handlesArray, true, INFINITE);

        end = GetTickCount();
        for (int i = 0; i < numOfThreads; ++i) {
                pi += pThreads[i].threadPi;
                //cout << "i thread" << iterationsPerThread[i] << endl;
        }
        pi /= N;
        cout << setprecision(70) << "Пи = " << pi << endl;
        cout << "Время потрачено: " << (end - start) << " мс" << endl;
        for (int i = 0; i < numOfThreads; ++i) {
                CloseHandle(pThreads[i].hThread);
        }
}
DWORD WINAPI worker(LPVOID lpParameter) {
        dwTlsThreadIndex = (DWORD)lpParameter;
        pThreads[dwTlsThreadIndex].calculating = true;
        unsigned long int beginIndex = 0;
        unsigned long int endIndex = 0;
        while (nextBlock <= TOTAL_BLOCKS) {
                double intermediatePi = 0;
                beginIndex = pThreads[dwTlsThreadIndex].nextBlockIndex*BLOCK_SIZE;
                endIndex = (pThreads[dwTlsThreadIndex].nextBlockIndex + 1)*BLOCK_SIZE;
                if (endIndex > N)
                        endIndex = N;
                for (unsigned long int i = beginIndex; i < endIndex; i++) {
                        double xi = (i + 0.5) / N;
                        intermediatePi += 4 / (1 + xi*xi);
                }
                pThreads[dwTlsThreadIndex].threadPi += intermediatePi;
                pThreads[dwTlsThreadIndex].calculating = false;
                SuspendThread(pThreads[dwTlsThreadIndex].hThread);
                pThreads[dwTlsThreadIndex].nextBlockIndex = InterlockedExchangeAdd(&nextBlock,
1);//nextBlock++;
                //++ iterationsPerThread [dwTlsThreadIndex];
                pThreads[dwTlsThreadIndex].calculating = true;
        }
        pThreads[dwTlsThreadIndex].finished = true;
        return 0;
}
double OMP() {
        int maxThreads;

        cout << "Максимум потоков: ";
        cin >> maxThreads;

        omp_set_dynamic(0);
        omp_set_num_threads(maxThreads);
```

```
	double start = GetTickCount();
	double pi = 0;
#pragma omp parallel for schedule(dynamic, BLOCK_SIZE) reduction(+:pi)
	for (int i = 0; i < N; i++) {
		double xi = (i + 0.5) / N;
		pi += 4 / (1 + xi*xi);
	}
	double end = GetTickCount();

	pi /= N;
	cout << setprecision(60) << "Пи = " << pi << endl;
	cout << "Время потрачено: " << (end - start) << " мс" << endl;

	return pi;
}
```