





Product Solutions Resources Open Source Enterprise Pricing



Sign in

Sign up

xmrig / xmrig 

Public

 Notifications

 Fork 3.4k

 Star 8.7k

 Code

 Issues 578

 Pull requests 40

 Actions

 Projects

 Wiki

 Security

 Insights

xmrig / src / base / kernel / Platform\_win.cpp 



xmrig Move Platform.

8b3f2d8 · 5 years ago

 History

```
1  /* XMRig
2  * Copyright 2010      Jeff Garzik <jgarzik@pobox.com>
3  * Copyright 2012-2014 pooler      <pooler@litecoinpool.org>
4  * Copyright 2014      Lucas Jones <https://github.com/lucasjones>
5  * Copyright 2014-2016 Wolf9466    <https://github.com/OhGodAPet>
6  * Copyright 2016      Jay D Dee   <jayddee246@gmail.com>
7  * Copyright 2017-2018 XMR-Stak    <https://github.com/fireice-uk>, <https://github.com
8  * Copyright 2018      SChernykh   <https://github.com/SChernykh>
9  * Copyright 2016-2019 XMRig      <https://github.com/xmrig>, <support@xmrig.com>
10 *
11 *   This program is free software: you can redistribute it and/or modify
12 *   it under the terms of the GNU General Public License as published by
13 *   the Free Software Foundation, either version 3 of the License, or
14 *   (at your option) any later version.
15 *
16 *   This program is distributed in the hope that it will be useful,
17 *   but WITHOUT ANY WARRANTY; without even the implied warranty of
18 *   MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the
19 *   GNU General Public License for more details.
20 *
21 *   You should have received a copy of the GNU General Public License
22 *   along with this program. If not, see <http://www.gnu.org/licenses/>.
23 */
24
25
26 #include <algorithm>
27 #include <winsock2.h>
28 #include <windows.h>
29 #include <uv.h>
30
31
32 #include "base/io/log/Log.h"
33 #include "Platform.h"
34 #include "version.h"
35
36
37 #ifdef XMRIG_NVIDIA_PROJECT
38 #   include "nvidia/cryptonight.h"
39 #endif
40
41
42 #ifdef XMRIG_AMD_PROJECT
43 static uint32_t timerResolution = 0;
44 #endif
45
46
```

xmrig / src / base / kernel / Platform\_win.cpp

↑ Top

Code

Blame

204 lines (155 loc) · 4.99 KB

Raw



```
51
52     HMODULE ntdll = GetModuleHandleW(L"ntdll.dll");
53     if (ntdll ) {
54         RtlGetVersionFunction pRtlGetVersion = reinterpret_cast<RtlGetVersionFunction>(
55
56         if (pRtlGetVersion) {
57             pRtlGetVersion((ULONG)VERSTANDARD, &major, &minor, &build, &revision);
```


Files


da22b3e


Go to file


> cmake


> doc


>  res


▼  src


>  3rdparty


>  backend


▼  base


>  api


>  io


▼  kernel


>  config


>  interfaces


 Base.cpp


 Base.h


 Entry.cpp


 Entry.h


 Platform.cpp


 Platform.h


 Platform\_mac.cpp


 Platform\_unix.cpp


 Platform\_win.cpp


 Process.cpp


 Process.h


 Signals.cpp


 Signals.h


>  net


>  tools


 base.cmake


>  core


>  crypto


>  net


 App.cpp


 App.h


 App\_unix.cpp


 App\_win.cpp


 Summary.cpp


 Summary.h

 config.json

 donate.h

 version.h

 xmrig.cpp

 .aitignore

```
57         platformGetVersion((LPOSVERSIONINFO) &result),
58     }
59 }
60
61 return result;
62 }
63
64
65 char *Platform::createUserAgent()
66 {
67     const auto osver = winOsVersion();
68     constexpr const size_t max = 256;
69
70     char *buf = new char[max]();
71     int length = snprintf(buf, max, "%s/%s (Windows NT %lu.%lu", APP_NAME, APP_VERSION,
72
73     # if defined(__x86_64__) || defined(_M_AMD64)
74     length += snprintf(buf + length, max - length, "; Win64; x64) libuv/%s", uv_version
75     # else
76     length += snprintf(buf + length, max - length, ") libuv/%s", uv_version_string());
77     # endif
78
79     # ifdef XMRIK_NVIDIA_PROJECT
80     const int cudaVersion = cuda_get_runtime_version();
81     length += snprintf(buf + length, max - length, " CUDA/%d.%d", cudaVersion / 1000, c
82     # endif
83
84     # ifdef __GNUC__
85     length += snprintf(buf + length, max - length, " gcc/%d.%d.%d", __GNUC__, __GNUC_MI
86     # elif _MSC_VER
87     length += snprintf(buf + length, max - length, " msvc/%d", MSVC_VERSION);
88     # endif
89
90     return buf;
91 }
92
93
94 bool Platform::setThreadAffinity(uint64_t cpu_id)
95 {
96     if (cpu_id >= 64) {
97         LOG_ERR("Unable to set affinity. Windows supports only affinity up to 63.");
98     }
99
100     return SetThreadAffinityMask(GetCurrentThread(), 1ULL << cpu_id) != 0;
101 }
102
103
104 uint32_t Platform::setTimerResolution(uint32_t resolution)
105 {
106     # ifdef XMRIK_AMD_PROJECT
107     TIMECAPS tc;
108
109     if (timeGetDevCaps(&tc, sizeof(TIMECAPS)) != TIMERR_NOERROR) {
110         return 0;
111     }
112
113     timerResolution = std::min<uint32_t>(std::max<uint32_t>(tc.wPeriodMin, resolution),
114
115     return timeBeginPeriod(timerResolution) == TIMERR_NOERROR ? timerResolution : 0;
116     # else
117     return resolution;
118     # endif
119 }
120
121
122 void Platform::restoreTimerResolution()
123 {
124     # ifdef XMRIK_AMD_PROJECT
125     if (timerResolution) {
126         timeEndPeriod(timerResolution);
127     }
128     # endif
129 }
130
131
```

```
132  void Platform::setProcessPriority(int priority)
133  {
134      if (priority == -1) {
135          return;
136      }
137
138      DWORD prio = IDLE_PRIORITY_CLASS;
139      switch (priority)
140      {
141      case 1:
142          prio = BELOW_NORMAL_PRIORITY_CLASS;
143          break;
144
145      case 2:
146          prio = NORMAL_PRIORITY_CLASS;
147          break;
148
149      case 3:
150          prio = ABOVE_NORMAL_PRIORITY_CLASS;
151          break;
152
153      case 4:
154          prio = HIGH_PRIORITY_CLASS;
155          break;
156
157      case 5:
158          prio = REALTIME_PRIORITY_CLASS;
159          break;
160
161      default:
162          break;
163      }
164
165      SetPriorityClass(GetCurrentProcess(), prio);
166  }
167
168
169  void Platform::setThreadPriority(int priority)
170  {
171      if (priority == -1) {
172          return;
173      }
174
175      int prio = THREAD_PRIORITY_IDLE;
176      switch (priority)
177      {
178      case 1:
179          prio = THREAD_PRIORITY_BELOW_NORMAL;
180          break;
181
182      case 2:
183          prio = THREAD_PRIORITY_NORMAL;
184          break;
185
186      case 3:
187          prio = THREAD_PRIORITY_ABOVE_NORMAL;
188          break;
189
190      case 4:
191          prio = THREAD_PRIORITY_HIGHEST;
192          break;
193
194      case 5:
195          prio = THREAD_PRIORITY_TIME_CRITICAL;
196          break;
197
198      default:
199          break;
200      }
201
202      SetThreadPriority(GetCurrentThread(), prio);
203  }
```

