Using GetLogicalProcessorInformationEx to see the relationship between logical and physical processors

devblogs.microsoft.com/oldnewthing/20131028-00

October 28, 2013

Raymond Chen

Today's Little Program uses the `GetLogicalProcessorInformationEx` function to print the mapping of logical processors to physical processors, as well as the mapping of logical processors to packages. (A dual-core processor is a single package with two cores. If those cores are themselves dual-hyperthreaded, then you have four logical processors total.)

```
#define STRICT
#include <windows.h>
#include <stdio.h>
template<typename T>
T *AdvanceBytes(T *p, SIZE_T cb)
{
 return reinterpret_cast<T*>(reinterpret_cast<BYTE *>(p) + cb);
}
```

The `AdvanceBytes` helper function takes a typed pointer and adds a byte offset to it. This is just a typing-saver function.

```
class EnumLogicalProcessorInformation
{
public:
 EnumLogicalProcessorInformation(LOGICAL_PROCESSOR_RELATIONSHIP Relationship)
  : m_pinfoBase(nullptr), m_pinfoCurrent(nullptr), m_cbRemaining(0)
 {
  DWORD cb = 0;
  if (GetLogicalProcessorInformationEx(Relationship,
                                       nullptr, &cb)) return;
  if (GetLastError() != ERROR_INSUFFICIENT_BUFFER) return;
  m_pinfoBase =
   reinterpret_cast<SYSTEM_LOGICAL_PROCESSOR_INFORMATION_EX *>
                                      (LocalAlloc(LMEM_FIXED, cb));
  if (!m_pinfoBase) return;
  if (!GetLogicalProcessorInformationEx(Relationship,
                                        m_pinfoBase, &cb)) return;
  m_pinfoCurrent = m_pinfoBase;
  m_cbRemaining = cb;
 }
 ~EnumLogicalProcessorInformation() { LocalFree(m_pinfoBase); }
 void MoveNext()
 {
  if (m_pinfoCurrent) {
   m_cbRemaining -= m_pinfoCurrent->Size;
   if (m_cbRemaining) {
    m_pinfoCurrent = AdvanceBytes(m_pinfoCurrent,
                                  m_pinfoCurrent->Size);
   } else {
    m_pinfoCurrent = nullptr;
   }
  }
 }
 SYSTEM_LOGICAL_PROCESSOR_INFORMATION_EX *Current()
                                          { return m_pinfoCurrent; }
private:
 SYSTEM_LOGICAL_PROCESSOR_INFORMATION_EX *m_pinfoBase;
 SYSTEM_LOGICAL_PROCESSOR_INFORMATION_EX *m_pinfoCurrent;
 DWORD m_cbRemaining;
};
```

Enumerating logical processor information is complicated due to the variable-size structures, so I wrap it inside this helper enumerator class.

Construct it with the relationship you are interested in, then use `Current()` to see the current item and `MoveNext()` to move to the next item. When there are no more items, `Current()` returns `nullptr`.

The constructor does the standard two-step query we've seen before: Ask for the required buffer size, then allocate a buffer, then ask for the buffer to be filled in. There is a TOCTTOU race condition if a processor is added dynamically, but I'm going to ignore that case because

this is a Little Program.

Since the `SYSTEM_LOGICAL_PROCESSOR_INFORMATION_EX` structure is variable-sized, walking the packed array is not a simple array indexing operation. Instead, you have to bump the pointer by the `Size` of the current element to find the next element.

Next comes a helper function to print processor affinity bitmasks.

```
void PrintMask(KAFFINITY Mask)
{
 printf(" [");
 for (int i = 0; i < sizeof(Mask) * 8; i++) {
  if (Mask & (static_cast<KAFFINITY>(1) << i)) {
   printf(" %d", i);
  }
 }
 printf(" ]");
}
```

Nothing exciting there.

Finally, we wrap it up inside a sample program that enumerates the cores and then, just for fun, enumerates the packages.

```
int __cdecl main(int argc, char **argv)
{
 for (EnumLogicalProcessorInformation enumInfo(RelationProcessorCore);
      auto pinfo = enumInfo.Current(); enumInfo.MoveNext()) {
   PrintMask(pinfo->Processor.GroupMask[0].Mask);
   printf("\n");
 }
 for (EnumLogicalProcessorInformation enumInfo(RelationProcessorPackage);
      auto pinfo = enumInfo.Current(); enumInfo.MoveNext()) {
   printf("[");
   for (UINT GroupIndex = 0; GroupIndex < pinfo->Processor.GroupCount; GroupIndex++)
{
    PrintMask(pinfo->Processor.GroupMask[GroupIndex].Mask);
   }
   printf(" ]\n");
 }
 return 0;
}
```

Enumerating processor cores produces a bunch of `PROCESSOR_RELATIONSHIP` structures, each with a single group that describes the logical processors assigned to the core.

Enumerating processor packages produces a bunch of `PROCESSOR_RELATIONSHIP` structures, and each one contains as many groups as there are cores in the package.

Bonus chatter: The CoreInfo utility from Sysinternals is a command-line tool that is a fancier version of this Little Program.

Raymond Chen

Follow