Using fibers to simplify enumerators, part 4: Filtering

devblogs.microsoft.com/oldnewthing/20050103-00

January 3, 2005

Raymond Chen

One type of higher-order enumeration is filtering, where one enumerator takes the output of another enumerator and removes some elements.

In a producer-driven enumerator, you would implement filtering by substituting a new callback function that responds to callbacks on behalf of the client for items that should be filtered, and forwarding callbacks to the client for items that are not filtered.

In a consumer-driven enumerator, you would implement composition by wrapping the enumerator inside another enumerator which drives the inner enumerator and forwards items that it wishes the caller to see.

A fiber-based enumerator behaves more like a consumer-driven enumerator, but,with easier state management.

Let’s write a filter enumerator that removes all directories and suppresses recursing into them.

```
class FilteredEnumerator : public FiberEnumerator {
public:
 FilteredEnumerator(LPCTSTR pszDir) : m_e(pszDir) { }
 LPCTSTR GetCurDir()
    { return m_e.GetCurDir(); }
 LPCTSTR GetCurPath()
    { return m_e.GetCurPath(); }
 const WIN32_FIND_DATA* GetCurFindData()
    { return m_e.GetCurFindData(); }
private:
 void FiberProc();
private:
 DirectoryTreeEnumerator m_e;
};
void FilteredEnumerator::FiberProc()
{
 FEFOUND fef;
 while ((fef = m_e.Next()) != FEF_DONE) {
  FERESULT fer;
  if (fef == FEF_DIR) {
   fer = FER_SKIP; // don't recurse into directories
  } else {
   fer = Produce(fef);
  }
  m_e.SetResult(fer);
 }
}
```

To produce items from this filtered enumerator, we run the real enumerator (`m_e`) and remove all directories, preventing them from being propagated to the filter's consumer and just responding "skip it" to the real enumerator.

You can test out this filtered enumerator with the same `TestWalk` function we've been using for the past few days. The only change you'll need to make is to the `main` function:

```
int __cdecl main(int argc, char **argv)
{
 ConvertThreadToFiber(NULL);
 FilteredEnumerator e(TEXT("."));
 TestWalk(&e);
 ConvertFiberToThread();
 return 0;
}
```

Observe that the program no longer recurses into subdirectories. It just tallies the sizes of the files in the current directory.

Next time, composition.

Raymond Chen

Follow