```
         break                        // Ran out of waves
      endif

      Wave w = $name
      if (index == 0)                 // Is this the first wave?
         Display w
      else
         AppendToGraph w
      endif
      index += 1
   while (1)          // Loop until break above
End
```

To make a graph of all of the waves in the current data folder, you could execute

```
DisplayWaveList(WaveList("*", ";", ""))
```

Operating on the Traces in a Graph

In a previous section, we showed an example that operates on the waves displayed in a graph. It used a wave reference function, TraceNameToWaveRef. If you want to write a function that operates on *traces* in a graph, you would *not* use wave reference functions. That's because Igor operations that operate on *traces* expect *trace names*, not wave references. For example:

```
Function GrayOutTracesInGraph()
   String list = TraceNameList("", ";", 1)
   Variable index = 0
   do
      String traceName = StringFromList(index, list)
      if (strlen(traceName) == 0)
         break                        // No more traces.
      endif

      // WRONG: ModifyGraph expects a trace name and w is not a trace name
      WAVE w = TraceNameToWaveRef("", traceName)
      ModifyGraph rgb(w)=(50000,50000,50000)

      // RIGHT
      ModifyGraph rgb($traceName)=(50000,50000,50000)

      index += 1
   while(1)
End
```

Using a Fixed-Length List

In the previous examples, the number of waves in the list was unimportant and all of the waves in the list served the same purpose. In this example, the list has a fixed number of waves and each wave has a different purpose.

```
Function DoLineFit(list)
   String list          // List of waves names: source, weight

   // Pick out the expected wave names

   String sourceStr = StringFromList(0, list)
   Wave source = $sourceStr

   String weightStr = StringFromList(2, list)
   Wave weight = $weightStr

   CurveFit line source /D /W=weight
End
```

You could invoke this function as follows: