```
   Execute cmd
End
```

Imagine that you pass a wave named *wave 1* to this function. Using the old technique, the Execute operation would see the following text:

```
wave 1 += 1
```

Igor would generate an error because it would try to find an operation, macro, function, wave or variable named *wave*.

Using the new technique, the Execute operation would see the following text:

```
'wave 1' += 1
```

This works correctly because Igor sees 'wave 1' as a single name.

When you pass a string expression containing a simple name to the $ operator, the name must not be quoted because Igor does no parsing:

```
String name = "wave 1"; Display $name                 // Right
String name = "'wave 1'"; Display $name               // Wrong
```

However, when you pass a *path* in a string to $, any liberal names in the path must be quoted:

```
String path = "root:'My Data':'wave 1'"; Display $path// Right
String path = "root:My Data:wave 1"; Display $path    // Wrong
```

For further explanation of liberal name quoting rules, see **Accessing Global Variables and Waves Using Liberal Names** on page IV-68.

Some built-in string functions return a list of waves. WaveList is the most common example. If liberal wave names are used, you will have to be extra careful when using the list of names. For example, you can't just append the list to the name of an operation that takes a list of names and then execute the resulting string. Rather, you will have to extract each name in turn, possibly quote it and then append it to the operation.

For example, the following will work if all wave names are standard but not if any wave names are liberal:

```
Execute "Display " + WaveList("*", ",", "")
```

Now you will need a string function that extracts out each name, possibly quotes it, and creates a new string using the new names separated by commas. The PossiblyQuoteList function in the WaveMetrics procedure file Strings as Lists does this. The preceding command becomes:

```
Execute "Display " + PossiblyQuoteList(WaveList("*", ",", ""), ",")
```

To include the Strings as Lists file in your procedures, see **The Include Statement** on page IV-166.

For details on using liberal names in user-defined functions, see **Accessing Global Variables and Waves Using Liberal Names** on page IV-68.

These functions are useful for programmatically generating object names: **CreateDataObjectName**, **CheckName**, **CleanupName**, **UniqueName**.

Programming with Data Folders

For general information on data folders, including syntax for using data folders in command line operations, see Chapter II-8, **Data Folders**.

Data folders provide a powerful way to organize data and reduce clutter. However, using data folders introduces some complexity in Igor programming. The name of a variable or wave is no longer sufficient to uniquely identify it because the name alone does not indicate in which data folder it resides.

If a procedure is designed to work on the current data folder, then it is the user's responsibility to set the current data folder correctly before running the procedure. When writing such a procedure, you can use