where *variable* is the name of a numeric variable containing the file reference number for the file to print to and the remaining parameters are as for printf. You get the file reference number using the Open operation, described under **Open and Close Operations** on page IV-196.

For debugging purposes, if you specify 1 for the file reference number, Igor prints to the history area instead of to a file, as if you used printf instead of fprintf.

wfprintf Operation

The wfprintf operation is similar to printf except that it prints the contents of one to 100 waves to a file. The syntax of the wfprintf operation is:

```
wfprintf variable, format [/R=(start,end)] wavelist
```

variable is the name of a numeric variable containing the file reference number for the file to print to.

Unlike printf, which rounds, wfprintf converts floating point values to integers by truncating, if you use an integer conversion specification such as “%d”.

Example Using fprintf and wfprintf

Here is an example of a command sequence that creates some waves and put values into them and then writes them to an output file with column headers.

```
Make/N=25 wave1, wave2, wave3
wave1 = 100+x; wave2 = 200+x; wave3 = 300+x
Variable f1
Open f1
fprintf f1, "wave1, wave2, wave3\r"
wfprintf f1, "%g, %g, %g\r" wave1, wave2, wave3
Close f1
```

This generates a comma delimited file. To generate a tab delimited file, use:

```
fprintf f1, "wave1\twave2\twave3\r"
wfprintf f1, "%g\t%g\t%g\r" wave1, wave2, wave3
```

Since tab-delimited is the default format for wfprintf, this last command is equivalent to:

```
wfprintf f1, "" wave1, wave2, wave3
```

Client/Server Overview

An application can interact with other software as a server or as a client.

A server accepts commands and data from a client and returns results to the client.

A client sends commands and data to a server program and receives results from the server.

For the Macintosh, see **Apple Events** on page IV-261 for server information and **AppleScript** on page IV-263 for client capabilities.

For Windows, see **ActiveX Automation** on page IV-265. Igor can play the role of an Automation server but not an Automation client. However it is possible to generate script files that allow Igor to indirectly play the role of client.

Apple Events

This topic is of interest to *Macintosh* programmers. For Windows, see **ActiveX Automation** on page IV-265.

This topic contains information for Igor users who wish to control Igor from other programs (e.g., AppleScript). It also contains information useful to people who are writing their own programs and wish to use Igor Pro as a computing or graphing engine.

There is also a mechanism that allows Igor to act like a controller and initiate Apple event communication with other programs. See **AppleScript** on page IV-263.

Apple Event Capabilities

Igor Pro supports the following Apple events:

Event	Suite	Action
Open Application	Required	Basically a nop; don't use.
Open Document	Required	Loads an experiment.
Print Document	Required	NA; don't use.
Quit Application	Required	Quits.
Close	Core	Acts on experiment, window or PPC port.
Save	Core	Acts on experiment only.
Open	Core	NA; don't use.
Do Script	Misc	Executes commands; can return ASCII results.
Eval Expression	Misc	Same as Do Script; obsolete but included for compatibility.

Apple Events — Basic Scenario

You use the Open Document event to cause Igor to load an experiment with whatever goodies you find useful. You then use the Do Script or Eval Expression events to send commands to Igor for execution and to retrieve results. To get data into Igor you write files and then send commands to Igor to load the data. To get data waves from Igor you do the reverse. To get graphics, you send commands to Igor that cause it to write a graphics file that you can read. You may then close the experiment and start over with a new one. You will not likely use the Save event.

Apple Events — Obtaining Results from Igor

To return information from Igor, you will need to embed special commands in the script you send to Igor for execution. When Igor encounters these commands, it appends results to a packet that is returned to your application after script execution ends. The special commands are variations on the standard Igor commands FBinWrite and fprintf. Both of these commands take a file reference number as a parameter. If the magic value zero is used rather than a real file reference number, then the data that would normally be written to a file is appended to the result packet.

As far as Igor is concerned, there is no difference between the Do Script and Eval Expression events. However, old applications may expect results from Eval Expression and not from Do Script.

To use waves and graphics with Apple events, you will need to write or read the data via standard Igor files. For example, you might include

```
SavePICT /E=-8 /P=MyPath as "Graphics.pdf"
```

in a script that you send to Igor for execution. You could then read the file in your application.

Apple Event Details

This information is intended for programmers familiar with Apple events terminology.

Some of the following events can act on experiments or windows.

To specify an experiment, use object class cDocument (`'docu'`) and specify either formAbsolutePosition with index=1 or formName with name=*name of experiment*.