Chapter
II-3

Experiments, Files and Folders

Experiments

An **experiment** is a collection of Igor objects, including waves, variables, graphs, tables, page layouts, notebooks, control panels and procedures. When you create or modify one of these objects you are modifying the **current experiment**.

You can save the current experiment by choosing File→Save Experiment. You can open an experiment by double-clicking its icon on the desktop or choosing File→Open Experiment.

Saving Experiments

There are two formats for saving an experiment on disk:

- As a packed experiment file. A packed experiment file has the extension .pxp.

  All waves, procedure windows, and notebooks are saved packed into the experiment file unless you explicitly save them separately.
- As an experiment file and an experiment folder (unpacked format). An unpacked experiment file has the extension .uxp.

  All waves, procedure windows, and notebooks are saved in separate files.

The packed format is recommended for most purposes. The unpacked format is useful for experiments that include very large numbers of waves (thousands or more).

Saving as a Packed Experiment File

In the packed experiment file, all of the data for the experiment is stored in one file. This saves space on disk and makes it easier to copy experiments from one disk to another. *For most work, we recommend that you use the packed experiment file format*.

The folder containing the packed experiment file is called the **home folder**.

To save a new experiment in the packed format, choose Save Experiment from the File menu.

Saving as an Unpacked Experiment File

In the unpacked format, an experiment is saved as an **experiment file** and an **experiment folder**. The file contains instructions that Igor uses to recreate the experiment while the folder contains files from which Igor loads data. The experiment folder is also called the **home folder**.

The main utility of this format is that it is faster for experiments that contain very large numbers of waves (thousands or more). However the unpacked format is more fragile and thus is not recommended for routine use.

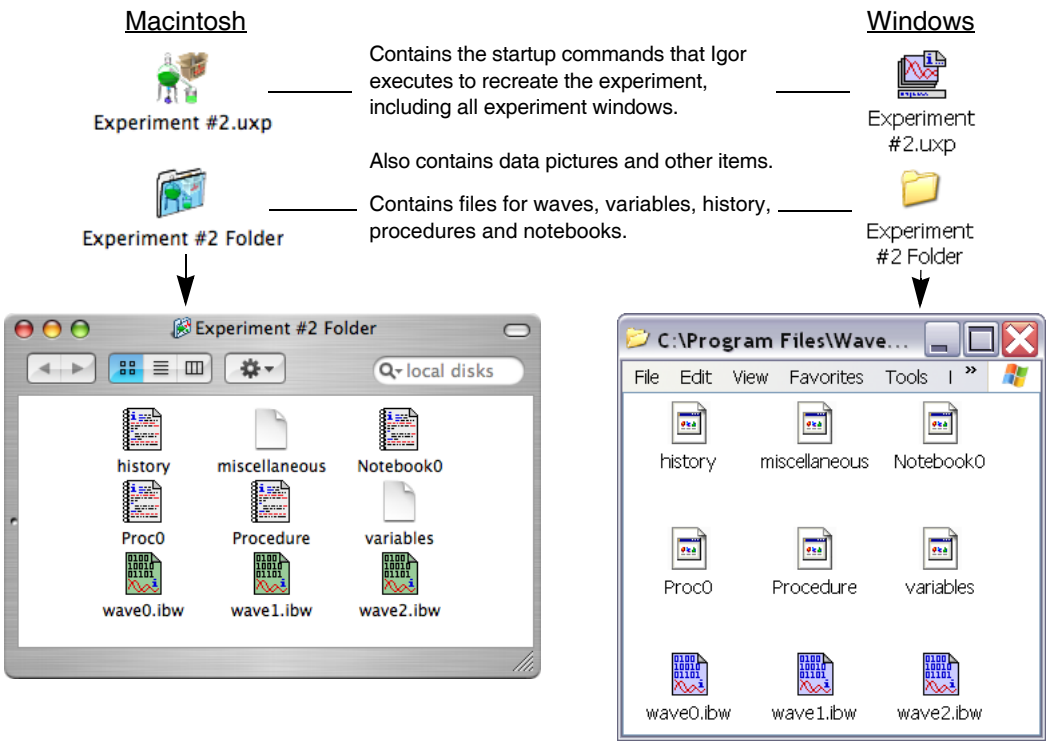
To save a new experiment in the unpacked format, choose Save Experiment from the File menu. At the bottom of the resulting Save File dialog, choose Unpacked Experiment Files from the popup menu. When you click Save, Igor writes the unpacked experiment file which as a ".uxp" extension.

Igor then automatically generates the experiment folder name by appending " Folder" or the Japanese equivalent, to the experiment file name. It then creates the unpacked experiment folder without further interaction. For example, if you enter "Test.uxp" as the unpacked experiment file name, Igor automatically uses "Test Folder", or the Japanese equivalent, as the unpacked experiment folder name.

If a folder named "Test Folder" already exists then Igor displays an alert asking if you want to reuse the folder for the unpacked experiment.

If the automatic generation of the unpacked experiment folder name causes a problem for you then you can save an experiment with the names of your choice using the **SaveExperiment** /F operation.

This illustration shows the icons used with an unpacked experiment and explains where things are stored.



You normally have no need to deal with the files inside the experiment folder. Igor automatically writes them when you save an experiment and reads them when you open an experiment.

If the experiment includes data folders (see Chapter II-8, **Data Folders**) other than the root data folder, then Igor will create one subfolder in the experiment folder for each data folder in the experiment. The experiment shown in the illustration above contains no data folders other than root.

Note that there is one file for each wave. These are Igor Binary data files and store the wave data in a compact format. For the benefit of programmers, the Igor Binary file format is documented in Igor Technical Note #003.

The "procedure" file holds the text in the experiment's built-in procedure window. In this example, the experiment has an additional procedure window called Proc0 and a notebook.

The "variables" file stores the experiment's numeric and string variables in a binary format.

The advantages of the unpacked experiment format are:

- Igor can save the experiment faster because it does not need to update files for waves, procedures or notebooks that have not changed.
- You can share files stored in one experiment with another experiment. However, sharing files can cause problems when you move an experiment to another disk. See **References to Files and Folders** on page II-22 for an explanation.

The disadvantages of the unpacked experiment format are:

- It takes more disk space, especially for experiments that have a lot of small waves.
- You need to keep the experiment file and folder together when you move the experiment to another disk.

Opening Experiments

You can open an experiment stored on disk by choosing Open Experiment from the File menu. You can first save your current experiment if it has been modified. Then Igor presents the standard Open File dialog.

When you select an experiment file and click the Open button, Igor loads the experiment, including all waves, variables, graphs, tables, page layouts, notebooks, procedures and other objects that constitute the experiment.

Some people mistakenly think that Igor recreates an experiment by reexecuting its history. See **How Experiments Are Loaded** on page II-24 for the real story.

Merging Experiments

Normally Igor closes the currently opened experiment before opening a new one. But it is possible to merge the contents of an experiment file into the current experiment. This is useful, for example, if you want to create a page layout that contains graphs from two or more experiments. To do this, press Option (*Macintosh*) or Alt (*Windows*) and choose Merge Experiment from the File menu.

Note: *Merging experiments is an advanced feature that has some inherent problems and should be used judiciously*. If you are just learning to use Igor Pro, you should avoid merging experiments until you have become proficient. You may want to skim the rest of this section or skip it entirely. It assumes a high level of familiarity with Igor.

The first problem is that the merge operation creates a copy of data and other objects (e.g., graphs, procedure files, notebooks) stored in a packed experiment file. Whenever you create a copy there is a possibility that copies will diverge, creating confusion about which is the "real" data or object. One way to avoid this problem is to discard the merged experiment after it has served its purpose.

The second problem has to do with Igor's use of names to reference all kinds of data, procedures and other objects. When you merge experiment B into experiment A, there is a possibility of name conflicts.

Igor prevents name conflicts for data (waves, numeric variables, string variables) by creating a new data folder to contain the data from experiment B. The new data folder is created inside the current data folder of the current experiment (experiment A in this case).

For other globally named objects, including graphs, tables, page layouts, control panels, notebooks, Gizmo plots, symbolic paths, page setups and pictures, Igor renames objects from experiment B if necessary to avoid a name conflict.

During the merge experiment operation, Igor looks for conflicts between target windows, between window recreation macros and between a target window and a recreation macro. If any such conflict is found, the window or window macro from experiment B is renamed.

Because page layouts reference graphs, tables, Gizmo plots, and pictures by name, renaming any of these objects may affect a page layout. The merge experiment operation handles this problem for page layouts that are open in experiment B. It does not handle the problem for page layout recreation macros in experiment B that have no corresponding open window.

If there are name conflicts in procedures other than window recreation macros, Igor will flag an error when it compiles procedures after finishing the merge experiment operation. You will have to manually resolve the name conflict by removing or renaming conflicting procedures.

Procedure windows have titles but do not have standard Igor names. The merge experiment operation makes no attempt to retitle procedure windows that have the same title.

The contents of the main procedure window from experiment B are appended to the contents of the main procedure window for experiment A.

During a normal experiment open operation, Igor executes experiment initialization commands. This is not done during an experiment merge.

Each experiment contains a default font setting that affects graphs and page layouts. When you do an experiment merge, the default font setting from experiment B is ignored, leaving the default font setting for experiment A intact. This may affect the appearance of graphs and layouts in experiment B.

The history from experiment B is not merged into experiment A. Instead, a message about the experiment merge process is added to the history area.

The system variables (K0…K19) from experiment B are ignored and not merged into experiment A.

Although the merge experiment operation handles the most common name conflict problems, there are a number problems that it can not handle. For example, a procedure, dependency formula or a control from experiment B that references data using a full path may not work as expected because the data from experiment B is loaded into a new data folder during the merge. Another example is a procedure that references a window, symbolic path or picture that is renamed by the merge operation because of a name conflict. There are undoubtedly many other situations where name conflicts could cause unexpected behavior.

Reverting an Experiment

If you choose Revert Experiment from the File menu, Igor asks if you're sure that you want to discard changes to the current experiment. If you answer Yes, Igor reloads the current experiment from disk, restoring it to the state it was in when you last saved it.

New Experiments

If you choose New from the File menu, Igor first asks if you want to save the current experiment if it was modified since you last saved it. Then Igor creates a new, empty experiment. The new experiment has no experiment file until you save it.

By default, when you create a new experiment, Igor automatically creates a new, empty table. This is convenient if you generally start working by entering data manually. However, in Igor data can exist in memory without being displayed in a table. If you wish, you can turn automatic table creation off using the Experiment Settings category of the Miscellaneous Settings dialog (Misc menu).

Saving an Experiment as a Template

A template experiment provides a way to customize the initial contents of a new experiment. When you open a template experiment, Igor opens it normally but leaves it untitled and disassociates it from the template experiment file. This leaves you with a new experiment based on your prototype. When you save the untitled experiment, Igor creates a new experiment file.

Packed template experiments have ".pxt" as the file name extension instead of ".pxp". Unpacked template experiments have ".uxt" instead of ".uxp".

To make a template experiment, start by creating a prototype experiment with whatever waves, variables, procedures and other objects you would like in a new experiment. Then choose File→Save Experiment As, choose Packed Experiment Template or Unpacked Experiment Template from the file type pop-up menu, and save the template experiment.

You can convert an existing experiment file into a template file by changing the extension (".pxp" to ".pxt" or ".uxp" to ".uxt").

The Macintosh Finder's file info window has a Stationery Pad checkbox. Checking it turns a file into a stationery pad. When you double-click a stationery pad file, Mac OS X creates a copy of the file and opens the copy. For most uses, the template technique is more convenient.

Browsing Experiments

You can see what data exists in the current experiment as well as experiments saved on disk using the Data Browser. To open the browser, choose Data→Data Browser. Then click the Browse Expt button. See **Data Folders** on page II-99 for details.

Symbolic Paths

A symbolic path is an Igor object that associates a short name with a folder on a disk drive. You can use this short name instead of a full path to specify a folder when you load, open or save a file. A full path is a complete specification of the location of a folder on a disk drive, as illustrated in the next section.

Igor creates some symbolic paths automatically and you can also create symbolic paths.

New Symbolic Path Dialog

To access the New Symbolic Path dialog, choose New Path from the Misc menu. Use of this dialog is illustrated in the next section.

Symbolic Path Example

This example illustrate why you should use symbolic paths and how to use them. We assume that you have a folder full of text files containing data that you want to graph in Igor and that the organization of your hard disk is as follows:

```
hd or C:
   Users
      Jack
         Documents
            Data
               2016
                  May
                  June
                  July
```

Assume that we want to access files in the June folder.

To create a symbolic path for the folder:

1. Choose New Path from the Misc menu. This displays the New Symbolic Path dialog.
2. Enter Data in the Name field.
3. Click the Path button and locate the June folder.
4. Check the Overwrite checkbox.
5. Click Do It to create the symbolic path.

The NewPath command created by the dialog makes a symbolic path named Data which references:

```
hd:Users:Jack:Documents:Data:2016:June     (Macintosh)
C:\Users\Jack\Documents\Data\June           (Windows)
```

The command generated by the dialog uses Macintosh-style paths with colons separating components:

```
NewPath/O Data, "hd:Users:Jack:Documents:Data:2016:June"     // Macintosh
NewPath/O Data, "C:Users:Jack:Documents:Data:2016:June"      // Windows
```

The NewPath operation can also accept Windows-style paths with backslash characters, but this can cause problems and is not recommended. For details, see **Path Separators** on page III-401.

Once you have created it, you can select the Data path in dialogs where you need to choose a file. For example, in the Load Waves dialog, you can select Data from the Path list. You then click the File button and choose the file to be loaded. Igor generates a command like:

```
LoadWave /J /P=Data "Data1.txt"
LoadWave /J /P=Data "Data2.txt"
LoadWave /J /P=Data "Data3.txt"
```

These commands load data files from the June folder.

By using a symbolic path instead of the full path to the file to be loaded, you have isolated the location of the data files in one object - the symbolic path itself. This makes it easy to redirect commands or procedures that use the symbolic path. For example, you can re-execute the NewPath command replacing June with July:

```
NewPath/O Data, "hd:Users:Jack:Documents:Data:2016:July"    // Macintosh

NewPath/O Data, "C:Users:Jack:Documents:Data:2016:July"     // Windows
```

Now, if you re-execute the LoadWave commands, they will load data from July instead of June.

Isolating a specific location on disk in a symbolic path also simplifies life when you move from one user to another or from one machine to another. Instead of needing to change the full path in many commands, you can simply change the symbolic path.

Automatically Created Paths

Igor automatically creates a symbolic path named **Igor** which refers to the "Igor Pro 7 Folder". The Igor symbolic path is useful only in rare cases when you want to access a file in the Igor Pro folder.

Igor also automatically creates a symbolic path named **IgorUserFiles** which refers to the Igor Pro User Files folder - see Special Folders for details. The IgorUserFiles symbolic path was added in Igor Pro 7.00.

Igor also automatically creates the **home** symbolic path. This path refers to the home folder for the current experiment. For unpacked experiments, this is the experiment folder. For packed experiments, this is the folder containing the experiment file. For new experiments that have never been saved, home is undefined.

Finally, Igor automatically creates a symbolic path if you do something that causes the current experiment to *reference* a file not stored as part of the experiment. This happens when you:

- Load an Igor Binary file from another experiment into the current experiment
- Open a notebook file not stored with the current experiment
- Open a procedure file not stored with the current experiment

Creating these paths makes it easier for Igor to find the referenced files if they are renamed or moved. See **References to Files and Folders** on page II-22 for more information.

Symbolic Path Status Dialog

The Symbolic Path Status dialog shows you what paths exist in the current experiment. To invoke it, choose Path Status from the Misc menu.

The dialog also shows waves, notebook files, and procedure files referenced by the current experiment via a given symbolic path.

Kill Paths Dialog

The Kill Symbolic Paths dialog removes from the current experiment symbolic paths that you no longer need. To invoke the dialog, choose Kill Paths from the Misc menu.

Killing a path does nothing to the folder referenced by the symbolic path. It just deletes the symbolic path name from Igor's list of symbolic paths.

A symbolic path is in use — and Igor won't let you kill it — if the experiment contains a wave, notebook window or procedure window linked to a file in the folder the symbolic path points to.

References to Files and Folders

An experiment can *reference* files that are not stored with the experiment. This happens when you load an Igor binary data file which is stored with a different experiment or is not stored with any experiment. It also

happens when you open a notebook or procedure file that is not stored with the current experiment. We say the current experiment is *sharing* the wave, notebook or procedure file.

For example, imagine that you open an existing text file as a notebook and then save the experiment. The data for this notebook is in the text file somewhere on your hard disk. It is not stored in the experiment. What *is* stored in the experiment is a *reference* to that file. Specifically, the experiment file contains a command that will reopen the notebook file when you next reopen the experiment.

Note: When an experiment refers to a file that is not stored as part of the experiment, there is a potential problem. If you copy the experiment to an external drive to take it to another computer, for example, the experiment file on the external drive will contain a *reference* to a file on your hard disk. If you open the experiment on the other computer, Igor will ask you to find the referenced file. If you have forgotten to also copy the referenced file to the other computer, Igor will not be able to completely recreate the experiment.

For this reason, we recommend that you use references only when necessary and that you be aware of this potential problem.

If you transfer files between platforms file references can be particularly troublesome. See **Experiments and Paths** on page III-399.

Avoiding Shared Igor Binary Files

When you load a wave from an Igor Binary file stored in another experiment, you need to decide if you want to *share* the wave with the other experiment or *copy* it to the new experiment. Sharing creates a reference from the current experiment to the wave's file and this reference can cause the problem noted above. Therefore, you should avoid sharing unless you want to access the same data from multiple experiments *and* you are willing to risk the problem noted above.

If you load the wave via the Load Igor Binary dialog, Igor will ask you if you want to share or copy. You can use the Miscellaneous Settings dialog to always share or always copy instead of asking you.

If you load the wave via the LoadWave operation, from the command line or from an Igor procedure, Igor will *not* ask what you want to do. You should normally use LoadWave's /H flag, tells Igor to "copy the wave to home" and avoids sharing.

If you use the Data Browser to transfer waves from one experiment to another, Igor always copies the waves.

Adopting Notebook and Procedure Files

Adoption is a way for you to copy a notebook or procedure file into the current experiment and break the connection to its original file. The reason for doing this is to make the experiment self-contained so that, if you transfer it to another computer or send it to a colleague, all of the files needed to recreate the experiment will be stored in the experiment itself.

To adopt a file, choose Adopt Window from the File menu. This item will be available only if the active window is a notebook or procedure file that is stored separate from the current experiment *and* the current experiment has been saved to disk.

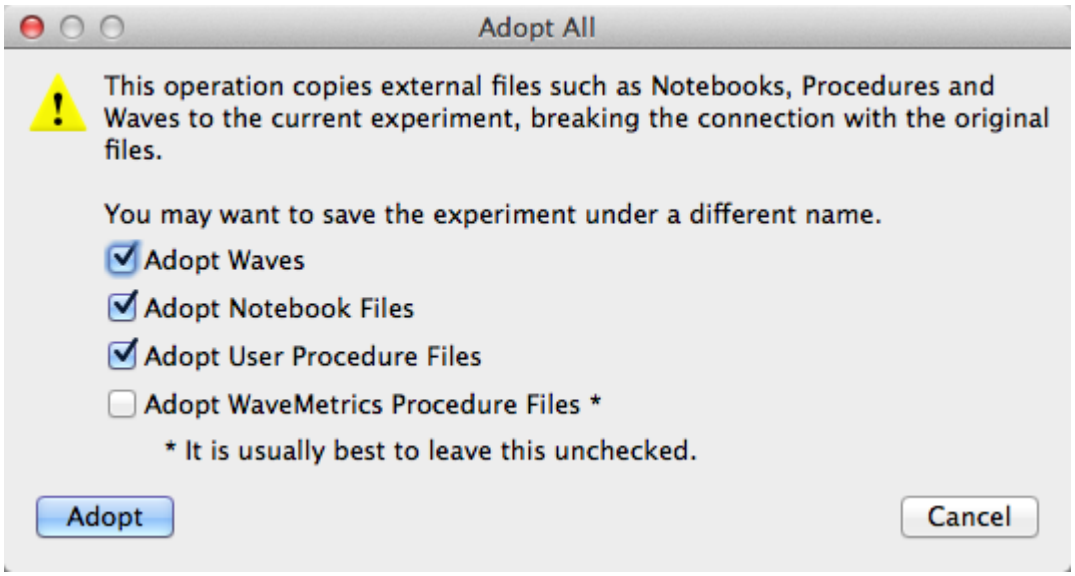
If the current experiment is stored in packed form then, when you adopt a file, Igor does a save-as to a temporary file. When you subsequently save the experiment, the contents of the temporary file are stored in the packed experiment file. Thus, the adoption is not finalized until you save the experiment.

If the current experiment is stored in unpacked form then, when you adopt a file, Igor does a save-as to the experiment's home folder. When you subsequently save the experiment, Igor updates the experiment's recreation procedures to open the new file in the home folder instead of the original file. Note that if you adopt a file in an unpacked experiment and then you do not save the experiment, the new file will still exist in the home folder but the experiment's recreation procedures will still refer to the original file. Thus, you should save the experiment after adopting a file.

To "unadopt" a procedure or notebook file, choose Save Procedure File As or Save Notebook As from the File menu.

Adopt All

You can adopt all referenced notebooks, procedure files and waves by pressing Shift and choosing File→Adopt All. This is useful when you want to create a self-contained packed experiment to send to someone else.



After clicking Adopt, choose File→Save Experiment As to save the packed experiment.

How Experiments Are Loaded

It is not essential to know how Igor stores your experiment or how Igor recreates it. However, understanding this may help you avoid some pitfalls and increase your overall understanding of Igor.

Experiment Recreation Procedures

When you save an experiment, Igor creates procedures and commands, called "experiment recreation procedures" that Igor will execute the next time you open the experiment. These procedures are normally not visible to you. They are stored in the experiment file.

The experiment file of an unpacked experiment contains plain text, but its extension is not ".txt", so you can't open it with most word processors or editors. You can open it by choosing File→Open File→Notebook and then selecting All Files from the file type pop-up menu. This is not something you would normally do, but it can be instructive.

As an example, let's look at the experiment recreation procedures for a very simple unpacked experiment:

```
// Platform=Macintosh, IGORVersion=7.000, ...

// Creates the home symbolic path
NewPath home ":Unpacked Experiment Folder:"

// Reads the experiment variables from the "variables" file
ReadVariables

// Loads the experiment's waves
LoadWave/C /P=home "wave0.ibw"
LoadWave/C /P=home "wave1.ibw"
LoadWave/C /P=home "wave2.ibw"
```

```
DefaultFont "Helvetica"

MoveWindow/P 5,62,505,335         // Positions the procedure window
MoveWindow/C 2,791,1278,1018      // Positions the command window

Graph0()                          // Recreates the Graph0 window

// Graph recreation macro for Graph0
Window Graph0() : Graph
   PauseUpdate; Silent 1   // building window...
   Display /W=(35,44,430,252) wave0,wave1,wave2
EndMacro
```

When you open the experiment, Igor reads the experiment recreation procedures from the experiment file into the procedure window and executes them. The procedures recreate all of the objects and windows that constitute the experiment. Then the experiment recreation procedures are removed from the procedure window and your own procedures are loaded from the experiment's procedure file into the procedure window.

For a packed experiment, the process is the same except that all of the data, including the experiment recreation procedures, is packed into the experiment file.

Experiment Initialization Commands

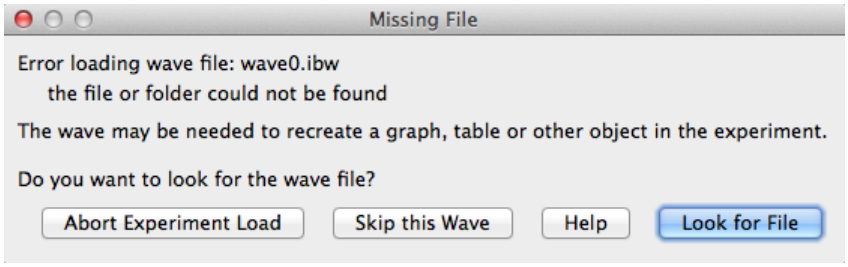
After executing the experiment recreation procedures and loading your procedures into the procedure window, Igor checks the contents of the procedure window. Any commands that precede the first macro, function or menu declaration are considered **initialization commands**. If you put any initialization commands in your procedure window then Igor executes them. This mechanism initializes an experiment when it is first loaded.

Savvy Igor programmers can also define a function that is executed whenever Igor opens any experiment. See **User-Defined Hook Functions** on page IV-264.

Errors During Experiment Load

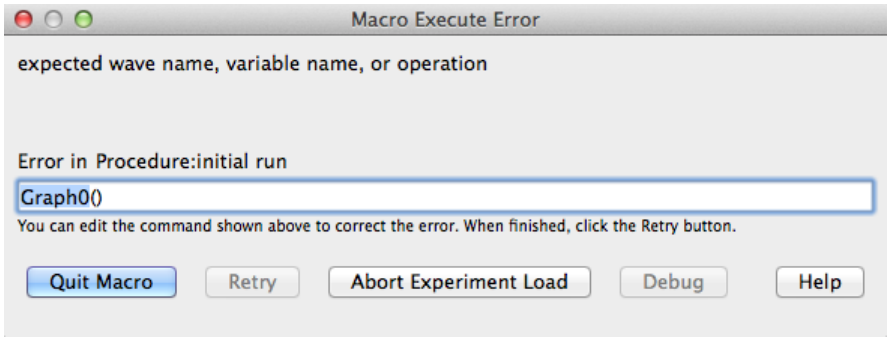
It is possible for the experiment loading process to fail to run to a normal completion. This occurs most often when you move or rename a file or folder and you can't help Igor find it. It also happens if you move an experiment to a different computer and forget to also move referenced files or folders. See **References to Files and Folders** on page II-22 for details.

These errors occur while Igor is executing the experiment recreation procedures. Igor uses several techniques to try to find the missing file or folder (see **How Igor Searches for Missing Folders** on page II-26). The techniques include asking you for help via a dialog like this:



If you elect to abort the experiment load, Igor will alert you that the experiment is in an inconsistent state. It displays some diagnostic information that might help you understand the problem and changes the experiment to Untitled. You should choose New Experiment or Open Experiment from the File menu to clear out the partially loaded experiment.

If you elect to skip loading a wave file, you may get another error later, when Igor tries to display the wave in a graph or table. In that case, you will see a dialog like this:



In this example, Igor is executing the Graph0 macro from the experiment recreation procedures in an attempt to recreate a graph. Since you elected to skip loading wave0, Igor can't display it.

You have three options at this point, as explained in the following table.

Option	Effect
Quit Macro	Stops executing the current macro but continues experiment load. In this example, Graph0 would not be recreated. After the experiment load Igor displays diagnostic information.
Abort Experiment Load	Aborts the experiment load immediately and displays diagnostic information.
Retry	In this example, you could fix the macro by deleting "wave0,". You would then click the Retry button. Igor would create Graph0 without wave0 and would continue the experiment load.

With the first two options, Igor leaves the experiment untitled so that you don't inadvertently wipe out the original experiment file by doing a save.

How Igor Searches for Missing Folders

When Igor saves an experiment, it stores commands in the experiment file that will recreate the experiment's symbolic paths when you reopen the experiment. The commands look something like this:

```
NewPath home ":Test Exp Folder:"
NewPath/Z Data1 "::Data Folder #1:"
NewPath Data2 "::Data Folder #2:"
NewPath/Z Data3 "hd:Test Runs:Data Folder #3:"      // Macintosh
NewPath/Z Data3 "C:Test Runs:Data Folder #3:"       // Windows
```

The location of the home folder is specified relative to the experiment file. The locations of all other folders are specified relative to the experiment folder or, if they are on a different volume, using absolute paths. Using relative paths, where possible, ensures that no problems will arise if you move the experiment file and experiment folder *together* to another disk drive or another location on the same disk drive.

The /Z flags indicate that the experiment does not need to load any files from the Data1 and Data3 folders. In other words, the experiment has symbolic paths for these folders but no files need to be loaded from them to recreate the experiment.

When you reopen the experiment, Igor executes these NewPath commands. If you have moved or renamed folders, or if you have moved the experiment file, the NewPath operation will be unable to find the folder. Here is what Igor does in this case.

If the symbolic path is not needed to recreate the experiment then Igor does nothing. It generates no error and just continues the load. The experiment will wind up without the missing symbolic path.

If the missing folder *is* needed to load some object then Igor searches for it using a number of techniques. The search uses additional information that Igor stores in the experiment file when the experiment is saved.

This includes such things as the full path to the folder and an "alias record", which are explained in the next section.

Folder Search Techniques

1. Search by full path

   A full path is one that starts from the volume that the folder is on. An example is "hd:Test Runs:Data Folder #3:" (*Macintosh*) or "C:\Test Runs\Data Folder #3\" (*Windows*).

   This technique will find the folder if the full path to the folder is the same as when the experiment was saved. This handles the case where you moved or renamed the experiment folder but did not move or rename the missing folder.
2. Search using Alias Manager (*Macintosh only*)

   The Alias Manager is a Macintosh feature designed to locate missing files and folders.
3. Search of the volume containing Igor

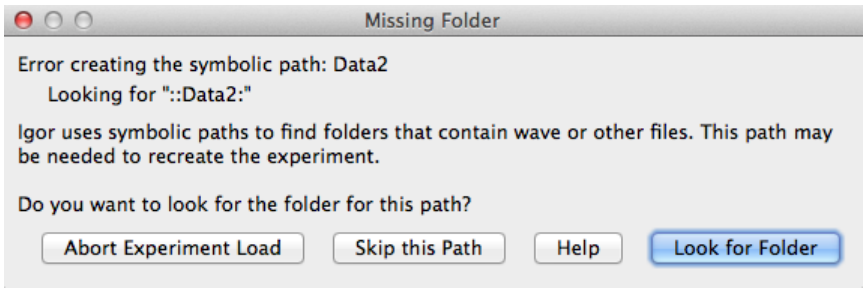
   If the path is a full path (e.g., "hd:Igor Work:Data Files" on Macintosh) then Igor searches for the folder at the same location but on the volume containing the Igor application (e.g., "C:\Igor Work\Data Files" on Windows).
4. Search of the volume containing the experiment file

   This is the same as the previous search except that Igor uses the volume containing the experiment file.

The last two techniques are designed to help find files when you move an experiment from one platform to another. They work only if the path is a full path, which will be the case if the target folder is on a different volume from the experiment file. If this is not the case, then the path will be relative to the experiment file and Igor will be able to find the target folder if it has the same relationship to the experiment file on both platforms.

If you use Igor on both Macintosh and Windows, it is best if you use the same folder hierarchy for your Igor files on both computers. This will give Igor the best chance of automatically finding missing folders.

If all of these techniques fail, Igor asks if you want to look for the folder by putting up a the Missing Folder dialog.



If you click Look for Folder, Igor presents another dialog in which you can find the missing folder.

If you click Skip this Path, Igor will not create the symbolic path and therefore you will get one or more errors later, when Igor tries to use it. For example, if the experiment loads two waves using the Data2 path then the experiment's recreation commands would contain two lines like this:

```
LoadWave/C/P=Data2 "wave0.bwav"
LoadWave/C/P=Data2 "wave1.bwav"
```

If you were unable to find the Data2 folder then each of these LoadWave commands will present the Missing Wave File dialog.

If you are unable to find the wave file and if the wave is used in a graph or table, you will get more errors later in the experiment recreation process, when Igor tries to use the missing wave to recreate the graph or table.

How Experiments Are Saved

When you save an experiment for the first time, Igor just does a straight-forward save in which it creates a new file, writes to it, and closes it. However, when you resave a pre-existing experiment, which overwrites the previous version of the experiment file, Igor uses a "safe save" technique. This technique is designed to preserve your original data in the event of an error during the save.

For purposes of illustration, we will assume that we are resaving an experiment file named "Experiment.pxp". The safe save proceeds as follows:

1. Write the new data to a temporary file named "Experiment.pxpT0". If an error occurs during this step, the save operation is stopped and Igor displays an error message.
2. Delete the original file, "Experiment.pxp".
3. Rename the temporary file with the original name. That is, rename "Experiment.pxpT0" as "Experiment.pxp".

On Windows, the temporary file name is "Experiment.pxpT0" but on Macintosh it is "Experiment.pxpT0.noindex". The ".noindex" suffix tells Apple's Spotlight program not to interfere with the save by opening the temporary file at an inopportune time.

The next three subsections are for use in troubleshooting file saving problems only. If you are not having a problem, you can skip them.

Experiment Save Errors

There are many reasons why an error may occur during the save of an experiment. For example, you may run out of disk space, the server volume you are saving to might be disconnected, or you may have a hardware failure, but these are uncommon.

The most common reason for a save error is that you cannot get write access to the file because:

1. The file is locked (Macintosh Finder) or marked read-only (Windows desktop).
2. You don't have permission to write to the folder containing the file.
3. You don't have permission to write to this specific file.
4. The file has been opened by another application. This could be a virus scanner, an anti-spyware program or an indexing program such as Apple's Spotlight.

Here are some troubleshooting techniques.

Macintosh File Troubleshooting

Open the file's Get Info window and verify that the file is not marked as locked. Also check the lock setting of the folder containing the file.

Next try doing a Save As to a folder for which you know you have write access, for example, to your home folder (e.g., "/Users/<user>" where <user> is your user name). If this works, the problem may be that you did not have sufficient permissions to write to the original folder or to the original file. Use the Finder Get Info window Sharing and Permissions section to make sure that you have read/write access for the file and folder.

If you are able to save a file to a new location but get an error when you try to resave the file, which overwrites the original file, then this may be an issue of another program opening the file at an inopportune time. This typically happens in step 3 of the safe-save technique described above. Try disabling your antivirus software. For a technical explanation of this problem, see http://developer.apple.com/qa/qa2006/qa1497.html.

Windows File Troubleshooting

Open the file's Properties window and uncheck the read-only checkbox if it is checked. Do the same for the folder containing the file.

Next try doing a Save As to a folder for which you know you have write access, for example, to your Documents folder. If this works, the problem may be that you did not have sufficient permissions to write to the original folder or to the original file. This would happen, for example, if the folder was inside the Program Files folder and you are not running as an administrator.

If you think you should be able to write to the original file location, you will need to investigate permissions. You may want to enlist the help of a local expert as this can get complicated and works differently in different versions of Windows.

If you are able to save a file to a new location but get an error when you try to resave the file, which overwrites the original file, then this may be an issue of another program opening the file at an inopportune time. This typically happens in step 3 of the safe-save technique described above. Try disabling your antivirus software. For a technical explanation of this problem, see http://support.microsoft.com/kb/316609.

Special Folders

This section describes special folders that Igor automatically searches when looking for help files, Igor extensions (plug-ins that are also called XOPs) and procedure files.

The folder containing the Igor Pro application file is called the "Igor Pro 7 Folder". Several subfolders in the Igor Pro 7 Folder are treated specially, as described below.

At launch time, Igor creates another special folder, called the Igor Pro User Files folder, outside of the Igor Pro 7 Folder. By default, this folder has the Igor Pro major version number in its name, for example, "Igor Pro 7 User Files", but it is generically called the "Igor Pro User Files" folder.

On Macintosh, the Igor Pro User Files folder is created by default in:

```
/Users/<user>/Documents/WaveMetrics
```

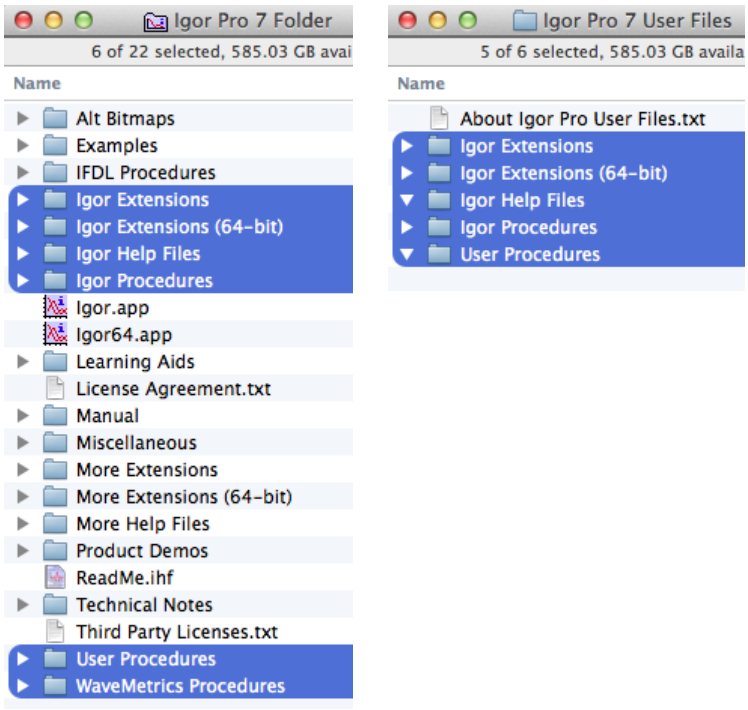
On Windows it is created by default in:

```
C:\Users\<user>\Documents\WaveMetrics
```

You can change the location of your Igor Pro User Files folder using Misc→Miscellaneous Settings but this should rarely be necessary.

Several subfolders in the Igor Pro User Files folder are also treated specially as described in the following sections.

Here we see the Igor Pro 7 Folder on the left and the Igor Pro User Files folder on the right with the special subfolders highlighted:



On Windows you will see a folder named IgorBinaries_Win32 instead of the Igor.app file and a folder named IgorBinaries_x64 instead of the Igor64.app file.

Help files, extensions and procedure files are active if they, or aliases or shortcuts pointing to them, are in the appropriate special subfolder. When you install Igor, special subfolders inside the Igor Pro 7 Folder contain files that are active. Examples include the standard WaveMetrics help files, standard WaveMetrics extensions, and standard WaveMetrics procedure files that add items to Igor's built-in menus.

You may want to activate additional help files, extensions and procedure files that are part of the Igor Pro installation, that you create, or that you receive from third parties. To do this, add files, or aliases/shortcuts pointing to files, to the special subfolders within the Igor Pro 7 User Files folder.

Do not add files to the special subfolders within the Igor Pro 7 folder. These folders are not accessible to non-administrative users and changing them complicates backup and updating Igor.

An exception to this rule applies if you are the administrator of a machine with multiple users and want to activate files for all users. In this case, you can add files to the special subfolders within the Igor Pro 7 folder.

Igor Pro 7 Folder

The Igor Pro 7 Folder is the folder containing the Igor application on Macintosh and the IgorBinaries folder on Windows. Igor looks inside the Igor Pro 7 Folder for these special subfolders:

```
Igor Help Files
Igor Extensions
Igor Extensions (64-bit)
Igor Procedures
User Procedures
WaveMetrics Procedures
```

The Igor installer puts files in the special folders. Igor searches them when looking for help files, extensions and procedure files. In most cases, you should not put files in these special folders - use the Igor Pro User Files folder instead.

Igor Pro User Files

The term “Igor Pro User Files” referens to the “Igor Pro X User Files” folder, where X is the Igor version.

Igor automatically creates the Igor Pro User Files folder at launch time if it does not already exist. Igor looks inside the Igor Pro User Files folder for these special subfolders:

```
Igor Help Files
Igor Extensions
Igor Extensions (64-bit)
Igor Procedures
User Procedures
```

You can put files or aliases/shortcuts pointing to files in these subfolders. Igor searches them when looking for help files, extensions and procedure files.

The default location of the Igor Pro User Files folder is:

```
Macintosh:
   /Users/<user>/Documents/WaveMetrics/Igor Pro 7 User Files

Windows:
   C:\Users\<user>\Documents\WaveMetrics\Igor Pro 7 User Files
```

You can change the location of your Igor Pro User Files folder using Misc→Miscellaneous Settings but this should rarely be necessary.

You can display the Igor Pro User Files folder on the desktop by choosing Help→Show Igor Pro User Files. To display both the Igor Pro 7 Folder and the Igor Pro User Files folder, press the shift key and choose Help→Show Igor Pro 7 Folder and User Files.

Igor Help Files Folder

When Igor starts up, it opens any Igor help files in "Igor Pro 7 Folder/Igor Help Files" and in "Igor Pro User Files/Igor Help Files". It treats any aliases, shortcuts and subfolders in "Igor Help Files" in the same way.

Standard WaveMetrics help files are pre-installed in "Igor Pro 7 Folder/Igor Help Files".

If there is an additional help file that you want Igor to automatically open at launch time, put it or an alias/shortcut for it in "Igor Pro User Files/Igor Help Files".

Igor Extensions Folder

An Igor extension, also called an XOP (“external operation”), is a plug-in that adds functionality to Igor. WaveMetrics provides some extensions with Igor. Igor users and third parties can also create extensions. See **Igor Extensions** on page III-450 for details.

When IGOR32 (the 32-bit version of Igor) starts up, it searches "Igor Pro 7 Folder/Igor Extensions" and "Igor Pro User Files/Igor Extensions" for 32-bit Igor extension files. These extensions are available for use in IGOR32. It treats any aliases, shortcuts and subfolders in "Igor Extensions" in the same way.

When IGOR64 (the 64-bit version of Igor) starts up, it searches "Igor Pro 7 Folder/Igor Extensions (64-bit)" and "Igor Pro User Files/Igor Extensions (64-bit)" for 64-bit Igor extension files. These extensions are available for use in IGOR64. It treats any aliases, shortcuts and subfolders in "Igor Extensions (64-bit)" in the same way.

Standard WaveMetrics extensions are pre-installed in "Igor Pro 7 Folder/Igor Extensions" and "Igor Pro 7 Folder/Igor Extensions (64-bit)".

Additional WaveMetrics extensions are described in the "XOP Index" help file, which you can access through the Help→Help Windows menu, and can be found in "Igor Pro 7 Folder/More Extensions" and "Igor Pro 7 Folder/More Extensions (64-bit)".

If there is an additional extension that you want to use, put it or an alias/shortcut pointing to it in "Igor Pro User Files/Igor Extensions" or "Igor Pro User Files/Igor Extensions (64-bit)".

Igor Procedures Folder

When Igor starts up, it automatically opens any procedure files in "Igor Pro 7 Folder/Igor Procedures" and in "Igor Pro User Files/Igor Procedures". It treats any aliases, shortcuts and subfolders in "Igor Procedures" in the same way. Such procedure files are called "global" procedure files and are available for use from all experiments. See **Global Procedure Files** on page III-353 for details.

Standard WaveMetrics global procedure files are pre-installed in "Igor Pro 7 Folder/Igor Procedures".

Additional WaveMetrics procedure files are described in the "WM Procedures Index" help file and can be found in "Igor Pro 7 Folder/WaveMetrics Procedures". You may also create your own global procedure files or obtain them from third parties.

If there is an additional procedure file that you want Igor to automatically open at launch time, put it or an alias/shortcut pointing to it in "Igor Pro User Files/Igor Procedures".

User Procedures Folder

You can load a procedure file from another procedure file using a #include statement. This technique is used when one procedure file requires another. See **Including a Procedure File** on page III-354 for details.

When Igor encounters a #include statement, it searches for the included procedure file in "Igor Pro 7 Folder/User Procedures" and in "Igor Pro User Files/User Procedures". Any aliases, shortcuts and subfolders in "User Procedures" are treated the same way.

If there is an additional procedure file that you want to include from your procedure files, put it or an alias/shortcut pointing to it in "Igor Pro User Files/User Procedures".

WaveMetrics Procedures Folder

The "Igor Pro 7 Folder/WaveMetrics Procedures" folder contains an assortment of procedure files created by WaveMetrics that may be of use to you. These files are described in the WM Procedures Index help file which you can access through the Help→Help Windows menu.

You can load a WaveMetrics procedure file from another procedure file using a #include statement. See **Including a Procedure File** on page III-354 for details.

There is no WaveMetrics Procedures folder in the Igor Pro User Files folder.

Activating Additional WaveMetrics Files

If you want to activate a WaveMetrics file that is stored in the Igor Pro 7 Folder, make an alias or shortcut for the file and put it in the appropriate subfolder of the Igor Pro User Files folder.

For example, the HDF5 file loader package consists of extensions named HDF5.xop and HDF5-64.xop, a help file named "HDF5 Help.ihf", and a procedure file named "HDF5 Browser.ipf". Here is how you would activate these files for use with Igor:

1. Press the shift key and choose Help→Show Igor Pro 7 Folder and User Files. This displays the Igor Pro 7 Folder and the Igor Pro User Files folder on the desktop.
2. Make an alias/shortcut for "Igor Pro 7 Folder/More Extensions/File Loaders/HDF5.xop" and put it in "Igor Pro User Files/Igor Extensions". This causes Igor to load the extension the next time IGOR32 is launched.
3. Make an alias/shortcut for "Igor Pro 7 Folder/More Extensions (64-bit)/File Loaders/HDF5-64.xop" and put it in "Igor Pro User Files/Igor Extensions (64-bit)". This causes Igor to load the extension the next time IGOR64 is launched.
4. This step is necessary only if you want the help file to be automatically opened. We usually skip this step and instead open the help file explicitly when we want it.

Make an alias/shortcut for "Igor Pro 7 Folder/More Extensions/File Loaders/HDF5 Help.ihf" and put it in "Igor Pro User Files/Igor Help Files". This causes Igor to automatically open the help file the next time Igor is launched.

5. Make an alias/shortcut for "Igor Pro 7 Folder/WaveMetrics Procedures/File Input Output/HDF5 Browser.ipf" and put it in "Igor Pro User Files/Igor Procedures". This causes Igor to load the procedure the next time Igor is launched and to keep it open until you quit Igor.
6. Restart Igor.

You can verify that the HDF5 extension and the HDF5 Browser procedure file were loaded by choosing Data→Load Waves→New HDF5 Browser.

Activating Other Files

You may create an Igor package or receive a package from a third party. You should store each package in its own folder in the Igor Pro User Files folder or elsewhere, at your discretion. You should not store such files in the Igor Pro 7 Folder because it complicates backup and updating.

To activate files from the package, create aliases/shortcuts for the package files and put them in the appropriate subfolder of the Igor Pro User Files folder.

If you have a single procedure file or a single Igor extension that you want to activate, you may prefer to put it directly in the appropriate subfolder of the Igor Pro User Files folder.

Activating Files in a Multi-User Scenario

Our recommendation is that you activate files using the special subfolders in the Igor Pro User Files folder, not in the Igor Pro 7 Folder. An exception to this is the multi-user scenario where multiple users are running the same copy of Igor from a server. In this case, if you want to activate a file for all users, put the file or an alias/shortcut for it in the appropriate subfolder of the Igor Pro 7 Folder. Users will have to restart Igor for the change to take effect.

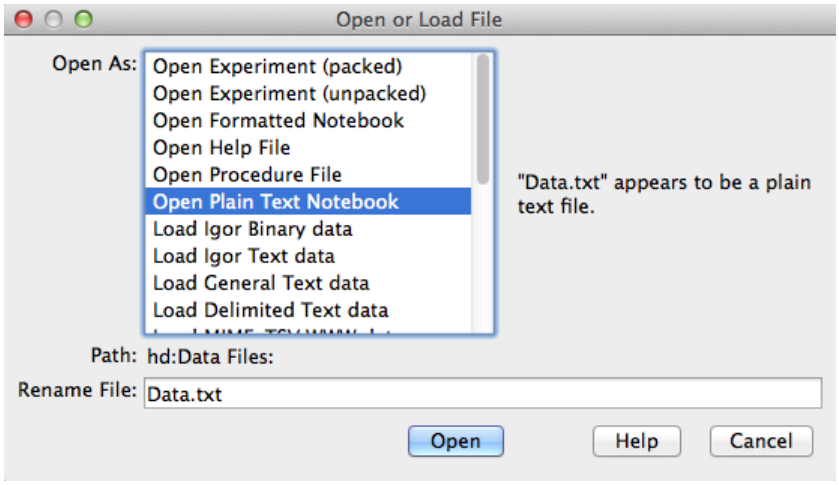
Igor File-Handling

Igor has many ways to open and load files. The following sections discuss some of the ways Igor deals with the various files it is asked to open.

Open or Load File Dialog

When you open a file using menu item, such as File→Open File→Notebook, there is no question of how Igor should treat the file. This is not always the case when you drop a file onto the Igor icon or double-click a file on the desktop.

Often, Igor can determine how to open or load a file, and it will simply do that without asking the you about it. Sometimes Igor recognizes that a file (such as a plain text file or a formatted Igor notebook) can be appropriately opened several ways, and will ask you what to do by displaying the Open or Load File Dialog. The dialog presents a list of ways to open the file (usually into a window) or to load it as data.



Tip: You can force this dialog to appear by holding down Shift when opening a file through the Recent Files or Recent Experiments menus, or when dropping a file onto the Igor icon.

This is especially useful for opening Igor help files as a notebook file for editing, or to open a notebook as a help file, causing Igor to compile it.

The list presents three kinds of methods for handling the file:

1. Open the file as a document window or an experiment.
2. Load the file as data without opening another dialog.
3. Load the file as data through the Load Waves Dialog or a File Loader Extension dialog.

If you choose one of the list items marked with an asterisk, Igor displays the selected dialog as if you had chosen the corresponding item from the Load Waves submenu of the Data menu.

Information about the file, or about how it was most recently opened, is displayed to the right of the list. The complete path to the file is shown below the list.

If you change the text in the Rename file edit box, Igor changes the file name before opening or loading the file.

Recent Files and Experiments

When you use a dialog to open or save an experiment or a file, Igor adds it to the Recent Experiments or Recent Files submenu in the File menu. When you choose an item from these submenus, Igor opens the experiment or file the same way in which you last opened or saved it.

For example, if you last opened a text file as an unformatted notebook, selecting the file from Recent Files will again open the file as an unformatted notebook. If you loaded it as a general text data file, Igor will load it as data again.

Igor does not remember all the details of how you originally load a data file, however. If you load a text data file with all sorts of fiddly tweaks about the format, Igor won't load it using the those same tweaks. To guarantee that Igor does load the data correctly, use the appropriate Load Data dialog.

Selecting an experiment or file while pressing Shift displays the Open or Load File dialog in which you can choose how Igor will open or load that file.

Desktop Drag and Drop

On Macintosh, you can drag and drop one or more files of almost any type onto the Igor icon in the dock or onto an alias for the Igor application on the desktop. On Windows, you must drag and drop files into an Igor window. One use for this feature is to load multiple data files at once.

If a dropped file was opened or loaded recently, it is listed in the Recent Files or Recent Experiments menu. In this case, Igor reopens or reloads it the same way.

If you press Shift while dropping a file on Igor, Igor displays the Open or Load File dialog in which you can choose how the file is to be handled.

Advanced programmers can customize Igor to handle specific types of files in different ways, such as automatically loading files with an XOP. See **User-Defined Hook Functions** on page IV-264.

IGOR64 Experiment Files

The 64 bit version of Igor can store waves larger than the 32-bit limit in packed experiment files but not in unpacked experiments. Huge wave storage uses a 64-bit capable wave record and is supported only for numeric waves.

The 32-bit versions of Igor, starting with version 6.20, supports 64-bit capable wave records. By default, IGOR32 attempts to read new these records and generates a lack of memory warning if they can not be loaded.

The rest of this section describes features for advanced users.

You can control some aspects of how Igor deals with 64-bit capable wave records using SetIgorOption. Remember that SetIgorOption settings last only until you quit Igor.

```
SetIgorOption UseNewDiskHeaderBytes = bytes
```

When bytes is non-zero, if the size of the wave data exceeds this value, the 64-bit capable wave record type is used. In IGOR32, the default value for bytes is 0 (off). In IGOR64, the default value for bytes is 100E6. If you want the 64-bit capable wave record type to be used only when absolutely necessary, use 2E9.

```
SetIgorOption UseNewDiskHeaderCompress = bytes
```

When bytes is non-zero, if the size of the wave data exceeds this value and if the 64-bit capable wave record type is used, the data is written compressed. The default value is 0 (off) because, although you can get substantial file size reduction for some kinds of data, there is also a substantial speed penalty when saving the experiment. In many cases, the compression actually results in a larger size than the original. If this occurs, Igor writes the original data instead of the compressed data.

```
SetIgorOption MaxBytesReadingWave = bytes
SetIgorOption BigWaveLoadMode = mode
```

These two options control how IGOR32 acts when loading experiments with the 64-bit capable wave record types.

mode can be 0 (default) for no special action. Igor attempts to read all the records.

If mode is 1, then any waves with data size exceeding bytes are silently skipped.

If mode is 2, then any waves with data size exceeding bytes are downgraded to a smaller size and only that portion of the data is loaded. Such partial waves are marked as locked and bit 1 of the lock flag is also set. In addition, the wave note contains the text "**PARTIAL LOAD**" and a warning dialog is presented after the experiment loads.

The default value for bytes is 500E6 in IGOR32.