# StuffIt

The Macintosh® Archive Utility

for the Apple® Macintosh

Written by Raymond Lau.

This document was set in Times® and Apple Courier with *MacWrite*®.

# 1. Preface/Introduction

## What is *StuffIt*?

*StuffIt* is a useful file archiving utility that allows you to do the following:
1. Gather many files together into a single "archive".
2. Optionally compress those files if desired.
3. Optionally encrypt those files if desired.
4. Optionally generate a report of the entries in an archive.
5. Segment large files for transmission by modem or for archiving from a hard disk onto floppies.

As a convenience, *StuffIt* can also:
6. Encode and Decode *BinHex4* .hqx files.
7. UnPack *PackIt II/III* ".pit" files and optionally convert them to the *StuffIt* format.

Of course, *StuffIt* also allows you to "extract" files that have previously been archived (or "stuffed") by you or another user.

## Why use *StuffIt*?

Gathering many files into an archive allows you to transmit many files as one over a modem or network. Compression of the files saves disk space and/or on-line time when transmitting or receiving files by disk or modem, respectively. Compression of rarely used files will allow you to better utilize available disk space. Encrypting files will protect your sensitive data from prying eyes.

## About the author:

*StuffIt* and its accompanying documentation were written by Raymond Lau with the help of a few other people. Work on *StuffIt* started around July-August, 1987 and has continued ever since. The author is devoted to continuously improving and updating *StuffIt* and its utilities, and he welcomes your suggestions, criticisms and other input. He may be reached at:

U.S.P.S.:
Raymond Lau
100-04 70 Ave.
Forest Hills, NY 11375-5133
United States of America

MacNET®: RayLau
GEnie™: RayLau
CompuServe®: 76174, 2617
Usenet: raylau@dasys1.UUCP
Delphi™: RaymondLau

## Distribution/Usage:

*StuffIt* is copyrighted in its entirety by Raymond Lau. It is being offered for distribution under the shareware philosophy. You may try *StuffIt* for 15 days. If you decide that you do not like it, please destroy your copy or pass it on to someone you know. If you decide to continue using it, you must register your copy by sending $20 to the author. Please include your complete name and address if it is not printed on the envelope or check and include the word *StuffIt* somewhere. If you would like to receive the next update, also include an extra $2 **or** a disk and a self-addressed-stamped-envelope. Be sure to indicate which version of the *StuffIt* you already have so I won't send you a duplicate. If you do not wish to use *StuffIt* but are finding that you need a program to extract files from archives created by others, you can use the freeware utility, *UnStuffIt*. This is so that you are not forced, and that you are not forcing others, into purchasing *StuffIt*. Please remember that your support is needed to make this method of distribution work. Improving *StuffIt* is of benefit to the Macintosh community in general, furthermore, it's enjoyable for me. However, without some financial support, I cannot pursue this avenue.

Those users updating from 1.40 to 1.5 may notice that the price has increased slightly. I feel that the added ability to handle Hierarchy Maintained Folders required a significant amount of effort on my part. I hope that this price change does not generate any resentment from within the Macintosh community.

Release versions of *StuffIt* may be distributed freely through electronic BBS', users' group libraries, timesharing services, public domain/shareware/freeware exchange forums and the like provided that such distribution is free except for normal duplication/distribution and/or usage charges. **Commercial distribution of any portion of the *StuffIt* "package", including such utilities as *UnStuffIt*, and the accompanying *StuffIt Utilities* require special licensing.**

## What you need to use *StuffIt*:

*StuffIt* works on any Macintosh computer with at least 512 kilobytes of memory. *StuffIt* requires that you use *System* 3.2 and the corresponding *Finder*™. For the Macintosh 512K, this is the latest *System* you can use. For the Macintosh Plus and larger machines, it is suggested that you use, as of this writing, *System* 4.2, which comes with the Universal System Update 5.0 or *System* 4.3, which comes with Universal System Update 6.0. To find out which *System* you have, select the *System* file of your startup disk in the *Finder* and pick Get Info. For more information on *System* files, consult the manuals which came with the computer and with the System update disks. If you are having difficulty, you can always to write to the author.

## What you need to know:

You should know generally how to operate your Macintosh, how to use the mouse and the menus, how to open documents, copy files, install applications onto your hard disks, etc. If you need help in this field, please consult the manuals which came with your Macintosh.

## What constitutes the *StuffIt* "package":

The *StuffIt* "package" consists of:
1. *StuffIt*, the application itself.
2. *UnStuffIt*, a freeware application which only extracts non-encrypted files from archives.
3. This documentation file.

If you are receiving *StuffIt* from the author on a disk, it will be accompanied by the *StuffIt Utilities* package and other miscellaneous programs. These are **not** part of the *StuffIt* "package". If you use them, please adhere to the requests included with them.

## Some terminology:

You should be familiar with the following terms when reading this document. Some of them have been used already.

<u>Archive</u> or <u>*StuffIt*</u> <u>archive</u> - A special file created by *StuffIt* which may contain other files, added by *StuffIt*. As a suggested convention, archive names usually end in ".sit".

<u>Compression</u> - The use of various algorithms in an attempt to compress files (reduce their size) before adding them into an archive.

Encryption - The use of an algorithm that makes the stored data in an archive unusable unless the proper password is entered. With encryption, you can keep your sensitive data from prying eyes.

Entry or an archive entry - A file which has been added into an archive by *StuffIt*.

File - Any file which can be accessed by your Macintosh computer.

Hierarchy Maintained Folder (HMF) - A special entry format which *StuffIt* 1.5 and later can understand. The HMF entry format allows you to add a whole folder to an archive, saving the structure of that folder. (Such as, what files are in what subfolders, etc.) Upon extraction, this structure will be restored. This entry format is not compatible with **older** versions of *StuffIt*.

LZW, Lempel-Ziv Welch, Lempel-Ziv, Huffman, and RLE - The names of some compression algorithms. You need not concern yourself with the details.

NewDE - A proprietary encryption technique used by *StuffIt* 1.4 and later. It is derived from the National Bureau of Standards' Data Encryption Standard and is believed to be extremely secure. Versions of *StuffIt* **before** 1.4 cannot understand this algorithm.

Read Only - Used in this document to refer to an archive which is locked or which resides on a "non-writeable" volume. You can only read information from these archives. You can't change the information contained in such archives and you can't add to them.

Volume - Any permanent storage device recognized by your Macintosh, such as a floppy disk, a hard disk, a tape, etc. In this document, volume may be used in context as the name of a volume. (The word "disk" is avoided because there are other devices such as CD-ROM and WORM that can be used as storage media.)

## Credits and Acknowledgements:

I would like to thank the following people and companies/organizations for their help with the development of *StuffIt*:

Richard Outerbridge - for providing an implementation of NewDE and help with optimizing the program in assembly.

David Schenfeld - for suggesting the idea of *StuffIt* in the first place and providing code from existing compression utilities.

William R. Whitford - for lending me various materials, like 2 sets of SIMMs, to ease the development process.

THINK Technologies Inc. - for providing the excellent *LightspeedC* development environment.

Leonard Rosenthol, Ray Terry and Bill Bumgarner - for help in testing the program in various stages of development.

J.W. McGuire - for designing many of the icons.

Alan Weber - for providing UnPacking code.

David L. Fulton of Fox Software - for providing a copy of their excellent *FoxBase+/Mac* database system, which is currently used to track registrations and other information.

The operators of BEC Public Access UNIX - for providing a UNIX system for my use and Usenet access.

All registered *StuffIt* users - for their support of the program and their comments.

# 2. Creating/Opening an Archive

## New Archives

Before working with a new archive, it must be created. Creating an archive is a very simple process. In *StuffIt*, go to the **File** menu and select **New Archive…** or type **[Command]-N**. You will be prompted with a dialog box asking for the name of the archive and the place where you want the archive to be created. This dialog works much like the Save As… dialogs in other applications. It is suggested that the archive name end in the letters ".sit" as a convention. It makes archive files easier to spot if everyone follows this convention, however, the name is totally arbitrary. Enter the archive name, select the destination volume/folder in the usual manner, and type return or click the New button.

### Checking Free Space

At the bottom of the dialog box, there's a line which shows how much space is free on a certain volume (disk). If you wish to check the amount of available free space on another volume, click on the **Free Space on Next Volume** button. Continuously clicking on this button will cycle through all mounted volumes and display the amount of free space on each volume.

(You can also install the *SFVol INIT* utility included with *StuffIt Utilities*. That provides a much more elegant solution)

## Opening Existing Archives

To work with an existing archive created by yourself or others, you must open it. Select **Open Archive…** from the **File** menu or type **[Command]-O**. You will be presented with the familiar Open dialog box as in other applications. Locate the archive you wish to open, select it and click on the Open button.

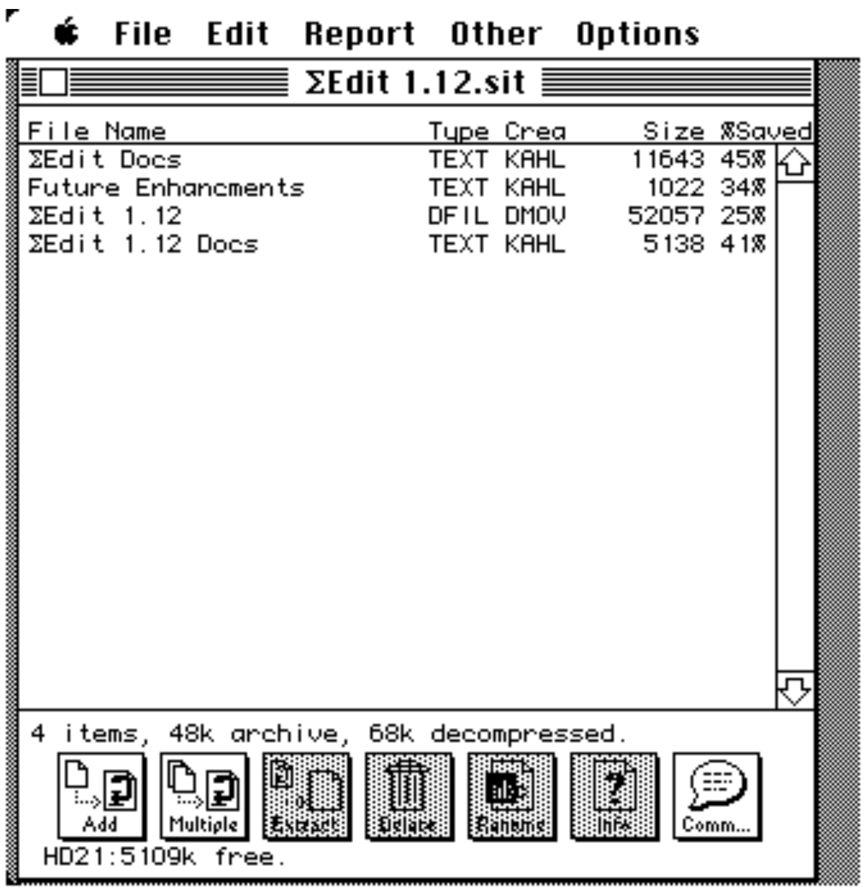
### Advanced Users

You are allowed to open archives created by *StuffIt* and compatible utilities as well as turnkeyed archives which do not have custom creators. If you hold down the **[Option]** key as you select **Open Archive…**, you can open **any** file. This is a **dangerous** feature. It allows you to open turnkeyed archives with custom creators **but** if you open a file that is not really an archive, you can irreparably damage it.

### Read Only Archives

If you open an archive on a locked volume or if you open a locked archive, you will not be allowed to make any changes to that archive. (You cannot add more files, delete entries, etc.) This is called a **read only** archive.

## Archive Contents Display



```
ΣEdit 1.12.sit

File Name                    Type Crea     Size %Saved
ΣEdit Docs                   TEXT KAHL    11643  45%
Future Enhancments           TEXT KAHL     1022  34%
ΣEdit 1.12                   DFIL DMOV    52057  25%
ΣEdit 1.12 Docs              TEXT KAHL     5138  41%

4 items, 48k archive, 68k decompressed.

HD21:5109k free.
```

After creating a new archive or opening an existing one, you will be presented with a window which has the name of the archive as its title. There will be a large rectangle which will contain the names of all entries in the archive, if any. (There won't be any entries if this is a new archive.)

If there are any file entries, the following information about the file entries should be displayed:
File Name, File Type, File Creator, Original File Size, Percentage Saved (if compressed)

If there are any HMF entries, the following information about the HMF entries should be displayed:
Folder Name (followed by a ':'), Original Folder Size, Percentage Saved (if compressed)

At the right edge of this rectangle is a scroll bar to scroll up and down if there are more entries than can be fitted in the provided rectangle.

Directly under the rectangle is a status line which tells you how many entries are in the archive, how much space the archive occupies on the volume, and how much space all the entries originally occupied before archival (which is exactly equal to the space the entries would take up if they were extracted and saved onto a volume)

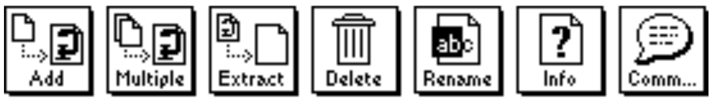
If the archive is a **read only** one, an icon of a padlock will follow at the end of this line.

Underneath is a collection of icons, an icon bar, which perform various functions that will be explained later.

At the very bottom is a line which indicates the name of the volume on which the archive resides and the amount of free space on that volume.

# 3. Working with Archives

There are quite a few operations which you can performed on archives. All of them can be accessed through the use of the icon bar. Some of them have keyboard shortcuts associated with them.



The icons, from left to right, are: Add, Multiple Add, Extract, Delete, Rename, Info and Comments. In the picture above, the Comments icon is showing a text filled "balloon". (The lines depict text.) This indicates that comments exist for the current archive. If the "balloon" is empty, no comments exist. If the archive is **read only**, the Add and Multiple Add icons will be dimmed. If no entry is selected, the Extract, Delete, Rename and Info icons will be dimmed.

The two main operations which can be performed on archives are adding files and extracting entries. First, we'll discuss three methods of selecting files to be added. This discussion assumes that the archive is not read only. Files cannot be added to read only archives.

## Adding Files

To add files to the current archive one at a time, click on the **Add** icon or type **[Command]-A**. You will be presented with a dialog box which resembles an Open dialog box, only slightly modified. The only differences are that it is larger, has a display line at the bottom, has two more buttons, one labeled Add and another labeled Set… and has a checkbox labeled Delete When Done.

**Open** - After selecting a file, clicking the Open button tells *StuffIt* to add that entry to the current archive, compressing and/or encrypting it if such is desired (more on this later). The phrasing of the word Open may seem anti-intuitive, but there is a reason for the separate Open and Add buttons. If a **folder** on a volume is selected, clicking Open will open that folder. ([Return] is identical to the Open button in function.)

**Add** - Add functions identically to Open **except** when a **folder** is selected. Add will have *StuffIt* add the **entire contents** of the folder and all sub-folders - one file at a time. (On a U.S. Keyboard, [Command]-[Return] is identical to the Add button.) Depending on your **Preferences…** settings, the folder may be added as one HMF or the contents of the folder will be added as individual files. (More on this in Chapter 4.)

**Tip**

It is possible to specify an entire volume to be added. (This may come in handy if you wish to add an entire floppy.) If no file or folder is selected (You can deselect any selected file/folder by clicking in the list and dragging, with the mouse button down, beyond the top of the list), clicking **Add** will specify the entire volume. This tip is particularly handy if you add the entire volume as a HMF. You can then later restore the entire volume exactly as it is except that everything will be in a folder with the name of the volume.

**Set…** - Next to the Set… button is an algorithms line which may display none, one or more of the following words: LZW, Huffman, Encrypt. Clicking on the Set button (or pressing [Command]-S) will allow you to change these algorithm settings. (You can also change them from the Options menu before entering the Add file dialog box.)

Below is an explanation of these algorithms:

LZW - An acronym for the Lempel-Ziv Welch compression algorithm. If this algorithm is specified, *StuffIt* will **attempt** to apply this compression algorithm to a file when added to an archive. LZW is generally effective for most files except for: sound files, files with a lot of bitmaps, already compressed dictionaries and some Microsoft applications (For the technically inclined, MS uses some kind of p-code which doesn't seem to compress well with LZW.). Because nothing is perfect in this world, LZW may not yield in a compressed file, in which case, *StuffIt* will just store the entry without any compression unless Huffman is also specified and proves to be more effective.

Huffman - Another file compression algorithm which generally works for most cases but is usually only superior to LZW in the cases where LZW performs poorly (specified above). In other cases, LZW usually outperforms Huffman.

**Note**

If both algorithms are checked on, *StuffIt* will attempt both and will use the one which yields the **best** results. **This is usually what you want!** However, there will always be cases where compression will fail altogether… Also, for files under 25K, if either or both compression algorithms are specified, a RLE technique will also be attempted. This attempt is rarely successful, but in some cases, it outperforms all others. It doesn't cost much time, so its presence is not much of a detriment.

Encrypt - If encryption is specified, a file that is added to the archive will also be **encrypted** with the NewDE encryption algorithm. This algorithm is believed to be extremely secure as it is derived from the National Bureau of Standards' Data Encryption Standard (DES). If you haven't entered a **password** for the archive yet (more on this later), you will be prompted to do so before the file is added.

**Delete When Done** - If this checkbox is **checked** (if there's an X through it), the file or folder being added will be automatically deleted if the add operation was successful. Note that this box unchecks itself each time. You must make a conscious effort of checking it.

## Adding Multiple Files

Compression may take a long time for larger files, even with *StuffIt.* I have toiled long and hard to make the program as fast as possible, but there are still limitations. The Mac only runs so fast. As a convenience, *StuffIt* allows you to specify up to 20 files and/or folders to be added in succession. This way, you can select all the big files, let *StuffIt* do its work and go away for lunch or whatever.

To use this feature, click on the **Multiple Add** icon or type **[Command]-M**. You will be presented with a small window in back of the standard add dialog box. Select the files/folders to add as you normally would. You will notice that they are appended to the Multiple Add list. When done, click on Done. You will be left with the Multiple Add list window, which has five buttons:

**Archive** ([Return]) - This choice tells *StuffIt* to go ahead and add all of the files on the list.

**Add** ([Command]-A) - Let's you add more files and/or folders to the Multiple Add list, if there are less than 20 entries on it.

**Delete** ([Command]-D) - If you have selected any files and/or folders on the Multiple Add list, you can have them removed from the list with this command.

**Note**

If you do not know how to select files on the list, refer to the selection procedure under the discussion of Extracting entries. The only difference is that the select all command is not available.

**Cancel** ([Command]-.) - Tells *StuffIt* that you don't want to add the files on the list after all.

**TEXT** ([Command]-T) - This brings you into the advanced TEXT add dialog, which will be described next.

## Advanced TEXT Add

An extension of the Multiple Add dialog allows you to specify files and/or folders to be added by typing in their pathnames. This may seem a silly idea in the Macintosh environment, and in general, it is! However, the textual interface allows you to specify many files through the use of **wildcard** characters. Beginners may skip this section if desired. Most people may never have the need for this feature, but if there's a need to archive a certain set of files having characteristic names, there is no substitute for this.

A wildcard character is a special character which can match any single character or any series of characters. There are two wildcard characters which *StuffIt* recognizes. They are ? and *. '?' will match any single character. '*' will match none, one, or more characters. With the use of wildcards and a pathname, it is possible to specify something like "all files ending in '.c' in the folder 'C' on the volume 'Disk1'". This translates into "Disk1:C:*.c<cr>" where <cr> is a carriage return (type the return key). Thus, as an example, if you ended all your MacPaint files with the suffix '.pntg', it's relatively easy to have *StuffIt* archive all of them on a given volume (and in a given folder, if you wish) with a wildcard such as [path name]:*.pntg.

Files and/or Folders are specified in the TEXT Add dialog as a list of full pathnames, each **ending in a carriage return.** (Each line **must** end with a carriage return and **must** be a full pathname!)

Pathnames are complete specifications of where a file is located.
The path portion includes the volume name and any folder names.
The filename portion is just that.

Ex: "MyVolume:MyFolder1:MyFolder2:MyFile"

Wildcards are allowed in the filename portion of pathnames. "?" will match a single character and "*" will match multiple characters or no characters at all.

Examples:
"abc" will only match "abc".
"a?c" will match "aac", "abc", "acc"
but <u>not</u> "accc" or "abb".
"a*c" will match "aac", "ac", "abbbbbc"
but <u>not</u> "ab", "abb", or "bbc" because the first
letter must be an "a" and the last letter is a "c".

Wildcards are allowed in the path portion of a pathname in a limited way. You can use an asterisk to indicate search all volumes in place of a volume name or you can use an asterisk before the filename portion to specify all subfolders.

Examples:
"*:System" will include all files named "System" on all volumes in the **root** (or disk) directory only.
"*:*:System" will include all files named "System" on all volumes in **any** folder.
"DD80:*:*.c" will include all files whose name ends in ".c" on the volume named DD80 in any folder.

You can match an "*" or a "?" by preceding it with a backslash, "\". To match a backslash, preceed it with another backslash.

Preceding each path name, you can specify which compression methods to try, whether to encrypt or not, and whether to delete or not by adding "::<flags><space>" where flags can be one or more of the following flag characters: L,l,H,h,E,e,D,d. L and l are for LZW; H and h are for Huffman; E and e are for encryption; and D and d are for auto-deletion of the files after they have been successfully added. Note that there is no way to add a HMF through this TEXT Add interface.

If you do not include these flags, the current flag selection will remain active.

Examples:
"::L MyDisk:MyFolder:*.a" will attempt to compress and archive, with the LZW compression algorithm, all files whose name ends in ".a" in the folder "MyFolder" on the volume "MyDisk".

Note that cAsE is <u>not</u> significant.

That is about all there is to the advanced TEXT add dialog/interface! This is not a feature for the beginner to struggle with. If you don't understand it but want to try it, that's fine, but let's exercise some restraint by not asking me too many questions on it. If you think you know how it works, but can't get it to work correctly, then go ahead and ask. If you're completely lost, don't worry about it. Come back to it at a later date.

## Special Note on Adding HMFs

Should a file system error occur when you add a HMF, *StuffIt* checks to see whether the error is considered to be **fatal** or not. (Fatal meaning an error that effectively prevents a HMF from being extracted properly) If a fatal error is detected, the partially added HMF entry will be deleted after posting an error alert box. If a non-fatal error is detected, you will be offered an option of either keeping the partially added HMF or deleting it. Note that if you decide to keep it, there is no provision for you to complete the add process.

## Extracting Files

Now that you have an archive with some entries in it, how do you recover those entries? Before you can use them you must **extract** (or decode, or unstuff, etc.) them from the archive. Once a file is extracted, it will be restored to its original status.

With an archive open, you will be looking at the contents display, as described earlier. In order to extract entries, you must **select** the entries first.

### Selecting Entries

The procedure for selecting entries is very important. You must select files before they can be extracted, deleted or renamed. (The latter two will be described later.) Although selection is done in the standard way of doing things on the Mac, I've received much mail asking about selecting more than one entry, etc.

To select a **single** entry, just click on it. If the entry is not in view, use the scroll bar in the usual manner to bring it into view and then click on it. You will see that the name will be displayed in inverse (background on foreground - or white on black).

To select a **continuous range** of entries, select the first entry in the range. Holding down the **[shift]** key, select the last entry in the range. All entries between and including the two will be selected and depicted in inverse.

To select a **discontinuous group** of entries (such as the first, the fourth and the tenth), select any one entry in the group. Holding down the **[command]** key, select the other entries.

Using the **[command]** key also allows you to de-select a selected entry or to select another not-yet-selected entry. (If you hold down the **[command]** key and click on an entry, it will be selected if it isn't already and deselected if it is.)

Once the entries you wish to extract are selected, click the **Extract...** icon (or type **[command]-E** or **double click** on a selected entry).

You will be presented with a dialog resembling the standard Save As... dialog box for each selected entry. Besides the usual Save button, there are also buttons labeled Save All, Skip and Cancel and the free space indicator/button which we have already discussed.

To simply save the extracted entry, click on the **Save** button or type return. Note that the original name will be given to you as the suggested name. You may change it before saving if you wish.

If the entry you are extracting is a HMF and if you are saving to a HFS destination, the name you enter will be the name of the upper-most folder of the HMF. The contents of the HMF will be stored within this upper-most folder and will have the exact same structure as the original folder (that is, the one that was originally added to the archive).

If you are saving a HMF to a MFS volume, the folder name will not be used. Any files in the HMF will be saved to the destination you specified. No folder structure will be restored. You will be warned of such an action. Should a duplicate file name be encountered, you will be asked to **rename** the file. Alternately, you can **skip** the file or **cancel** the operation all together.

To extract and save all selected entries to the same location using the suggested names, click on **Save All** or if you wish, on a U.S. keyboard, type **[Command]-<return>**.
Should a file name conflict arise during the save all process or should some error occur, you will be presented with this dialog again, allowing you to cancel or continue the extraction process.

## Extracting All Entries: A SHORTCUT

There are two short cuts to help you easily extract all entries in an archive. If you are opening the archive or archives from the *Finder*, hold down the **[shift]** key as *StuffIt* loads. All the archives you open will be processed in order. The contents of each archive will all be extracted and saved into the same folder which the archive originally resided in. The last archive will be left open. If you are opening an archive from within *StuffIt*, holding down the **[shift]** key as you click **Open**, **double-click**ing, or pressing **[return]**, after you have selected the archive you wish to open, will accomplish the same thing.

## Progress Indication/Aborting

We're almost through discussing the add/extract functions, but there's one more thing to be covered.

As you have noticed by now if you've tried adding/extracting, there are indicators which show some information about the files and the progress of the operation. Before we continue, you should be familiar with the concept of **forks** as it relates to Macintosh files. All files on the Mac consist of one or two (or none if it's an empty file) forks: the **resource fork** and the **data fork**. When *StuffIt* works with a file or an entry, it handles things one fork at a time.

When adding files, you will see the name of the file to be added, the algorithms you've requested to be used, the size of each fork, bar graph progress indicators for each fork and the compressed size, if applicable, obtained by each compression algorithm attempted. (If you did not want *StuffIt* to compress using an algorithm, it will just report the normal size in the appropriate space.) Of all the pieces of information here, most users will be concerned most about the bar graph indicator. It works more or less as expected except for one thing. If the LZW compression algorithm is specified and it proves to be successful, the latter part (one third to a half) of the bar graph is drawn almost instantly. Because of the nature of the LZW algorithm, almost all of the "work" is done in the first half. The second half is a very quick pass. If that didn't make sense, try this explanation: The bar graph is relative to the amount of disk reading *StuffIt* performs on each fork and not on the time needed to perform the operation. For LZW, the first half or so of the disk reading is a lot slower than the final pass, which is practically instantaneous.

If you are adding a folder, either as a HMF or as individual files, the folder name will be displayed on top of the above described display.

When extracting files, you will see some statistics about the entry along with bar graphs for each fork. Here, there are no surprises. The bar graph is a very accurate indication of the time remaining and the amount of disk reading left. (They are both the same, relatively speaking.)

If you are extracting a hierarchy maintained folder, the folder name will be displayed on top of the above described display.

During the add/extract processes, if you hold down the **[Command]-.** key combination (on a U.S. keyboard), *StuffIt* will **abort** the current operation. Keep holding down the two keys until you hear *StuffIt* beep at you.

That's it for adding/extracting files - the two primary functions of *StuffIt*. There are a few features to make it easier to work with archives and they will be discussed next.

## The Delete Operation

Sometimes, it would be very convenient to be able to delete an entry from an archive. Perhaps it was added erroneously or perhaps the entry is to be replaced by a newer file. *StuffIt* provides such a capability.

Select the entries you wish to delete. Click on the **Delete** icon or type **[Command]-D**. *StuffIt* will present you with a dialog asking you to confirm the deletion. Acknowledge and the entries will be deleted. To bypass the warning dialog, hold down the **[option]** key while issuing the delete command. (Of course, if the archive is read only, you cannot delete any entries from it…)

**Note**
Deleting may take some time as *StuffIt* has to move the entries around in the archive to

reclaim the vacated space. The larger your archive, the longer *StuffIt* may take. So, let's keep those archives down to a reasonable size and a reasonable number of files. (Caveat emptor (actually, a bit of trivia): *StuffIt* can handle up to 1000 entries and files/archives up to about 350 megabytes in length. I doubt if this limit will be exceeded in the near future… but who knows what will happen when they start putting stacks on giga-byte disks?)

## The Info Operation

If you would like to know some information on the entries in an archive, select the entries you wish to query and click on the **Info** icon or type **[Command]-I**. You will be presented with an info window for each selected entry. If the entry is a file, you will see the file's original size, compressed size, creator, type, creation/modification date, and if *StuffIt* can figure it out, the name of the creating application. If the entry is a HMF, you will see the total compressed and decompressed sizes of the HMF as well as a listing of the HMF's contents. You can use the scroll bar to go through the listing as needed. Just click **OK** or type a **[Return]** when you're done viewing the info for an entry. *StuffIt* will proceed to display info on the next entry you selected or if that was the last entry, *StuffIt* will return you to the archive contents display. At any point, you can also click **Cancel All** or type a **[Command]-.** and you will be returned to the archive contents display instantly. No info will be displayed on remaining selected entries.

## The Rename Operation

*StuffIt* displays the entries of an archive with the original names. When extracting, it will also suggest these as the default names. However, it is possible to change the file name without having to extract the entry, delete it, rename the extracted file and re-add it. Just select the entry or entries you wish to rename. Click on the **Rename** icon or type **[Command]-R**. *StuffIt* will cycle through each selected entry and give you the opportunity to rename it. (Of course, if the archive is read only, you can't rename entries!)

## Archive Comments

It is possible to store some short comments along with each archive. With the archive open, select **Comments…** from the **File** menu or click on the **Comments** icon. You will see the current archive comments, if any. You may change them or add them if desired. (Of course, if the archive is read only, you can't add or change comments!)

### Note

If the **Comments** icon shows an empty "balloon", no comments exists for the current archive. You may add some, of course. If the "balloon" is filled with lines which represent text, then comments exist for the current archive.

## Generating a Report

*StuffIt* includes the capability to generate a report, in the form of a 'TEXT' file, for the current archive. The report will list some statistical information about all the entries in the archive. Select **Generate Report** from the **Report** menu (or type **[Command]-G**). You will be presented with the familiar save as dialog box. Indicate the name of the report file and the location you wish to have it saved. Once a report is generated, it may be viewed with a text editor or a word processor. Tab spacing should be set to 8 or 16 spaces for optimal results. The report may also be imported into a spreadsheet such as Excel and possibly into a database.

# 4. Options/MultiFinder

Under the **Options** menu, besides the four algorithm options that we have already discussed, there is an item called “Preferences”. Selecting this item brings up a dialog box with a few radio buttons and checkboxes. The following is a description of what each option is:

**Restore Modification Dates**

If checked, *StuffIt* will restore the modification dates of files extracted from an archive. This is usually what you want unless you have a hard disk backup program which relies on modification dates for incremental back ups. In such a case, if the modification dates are restored, the program may not back up the extracted file.

**Sounds When Done**

When adding or extracting large files, you may want to abandon your lovely Mac for a while. With this option checked, *StuffIt* will beep for you when it’s done and ready for your commands.

**Allow Background Tasks**

If *MultiFinder*™ is running and this option is checked, *StuffIt* will delegate time to *MultiFinder* background tasks, such as the *Print Monitor*, during the add and extract processes. It is *possible* to click on the little icon at the upper right hand of the menu bar to make *StuffIt* run in the background, however, *StuffIt* is a time hog. It has a lot of computations to do when compressing and/or encrypting. Thus, it will give very **little** time to the new foreground task. If you have a speedy machine, it may be enough. If not, it probably won’t be.

**IMPORTANT**

Due to a bug in *MultiFinder* 1.0, which accompanied System Update 5.0, if *StuffIt* extracts a file into a folder that’s open on the desktop (or in any other way creates a file there, including the conversion of a *BinHex4* file, etc.), the file’s attributes will **not be set correctly**. This bug has been fixed in the version of *MultiFinder* which accompanies System Update 6.0. Work only with closed folders or use the newer *System*.

**Note**
While *StuffIt* is working with a file, or even with an archive, it is imperative that you do not move, rename, delete or otherwise alter the file. For example, if *StuffIt* is working with “Archive.sit”, and you move the file with the *Finder* (which shouldn’t be allowed, but I don’t trust things to chance.), *StuffIt* will get hopelessly confused and unexpected problems will arise.

**Folder Options**

The three options dealing with folders determines what *StuffIt* will do when asked to add a folder.

Always add as **individual files** means that *StuffIt* will never add a folder as a HMF. The folder’s contents will be added as separate files and each of these files will be a separate entry in an archive.

Always add as **folders (HMF)** means that the entire folder will be added as a HMF. The folder structure will be saved with the HMF. The HMF will be one entry in an archive.

**Ask the user** what to do means that *StuffIt* will put up a dialog box each time the you request a folder to be added. The dialog box will offer the you a choice between files and HMF. To select what you want, click on the appropriate button or type **F** for files or **H** for HMF.

# 5. More on Encryption

You should already know how to add files with encryption turned on and how to extract encrypted entries. However, there are a few more details you should know about *StuffIt*'s **NewDE** encryption algorithm.

NewDE is believed to be extremely secure and is reasonably fast. It was derived from the National Bureau of Standards' Data Encryption Standard (DES). Thus, I must warn you that if you **lose the password**, you're out of luck. Richard Outerbridge and myself are not miracle workers. We cannot magically recover encrypted entries if the password is gone.

You are allowed **one** password per archive. Currently, there is no option to change the password once entered. This will probably be added in the future. *StuffIt* saves a **seal** of the password. (A check value, sort of…) This seal is required for *StuffIt* to check that the password, entered at a later date, is valid and is needed to correctly decrypt entries. The seal does not allow a person to work backwards and recover the password. The odds are overwhelmingly against it. (Of course, if the person had a Cray supercomputer and a few months, he could do almost anything…)

**The password, once entered for any open archive, will remain loaded until you quit *StuffIt.*** This is designed to be a feature. If you have entries in many archives encrypted with the same password, once you enter it for the first archive, working with the other archives will **not** require you to reenter the password. Of course, if you try to extract an encrypted entry from an archive with a different password, you must enter that password (which will then remain loaded hence forth…) **This may pose a security risk if you are not aware of it.** Consider the following scenario: You open an archive from your hard disk and extract an encrypted entry. You will be prompted for the password. You enter the password and the entry is extracted successfully. You then remove the disk with the extracted file and leave your Mac unattended. Because the password is still loaded, some corporate spy may walk over and extract that entry and save it onto his disk without knowing the password! So, **quit when done!** I do not see this as a threat if you are made aware of the feature. The convenience of this feature far outweighs the risks, in my opinion.

# 6. Segmenting Files

*StuffIt* provides a general purpose facility for segmenting large files into parts (segments). You may want to do this in order to fit a large file from a hard disk onto floppies or to send a large file in chunks over a communications link. (Of course, you may have other reasons for segmenting files.)

Before you segment a file, select **Set Segment Size…** from the **Other** menu. You will be presented with a scroll bar that works in 10 kilobyte increments from 10k to 3200k. Set the size of each segment. (All segments will be all the same size, except for the last segment, which may be smaller.) A few buttons with some standard sizes are provided for your convenience.

Next, select **Segment…** from the same menu. You will be asked to select a file to segment. Do so. For each segment of the file, you will be prompted for the name and location (to save) for that segment. After the final segment is saved, the segment operation will be complete. You will have the segments of the original file. You must have all the segments to be able to recreate the original file at a later date or in a different location.

## Joining Segments

Now, if you have all the segments of a file, they must be **join**ed back together in order to recover the original file. Select the **Join…** command from the **Other** menu. You will be asked to find the first segment. Do so. You will be asked to find each of the other segments in sequence. Do so and the join operation will be complete. You will have the original unsegmented file.

# 7. UnPack

For compatibility purposes, *StuffIt* supports files created by the *PackIt II* and *PackIt III* compression utilities (as long as encryption wasn't used.) *StuffIt* is noticeably faster at UnPacking than *PackIt*.

To UnPack a .pit (*PackIt* files usually have the ".pit" suffix) file, select **UnPack** from the **Other** menu. Find the file you wish to UnPack. For each entry, you will be prompted as to what to name the file and where to save it. (Or if an archive is open, you can have it converted to the *StuffIt* format.) You can use **Save All** and **Convert All** to save all remaining entries using the default name to the same location or to convert all remaining entries, respectively. You may also skip an entry or entries. Note that due to a design oversight in *PackIt*, skipping requires the actual decompression of the entry and is thus not very fast.

*PackIt* was written by Harry Chesley.

# 8. BinHex4

If you don't know what *BinHex4* is, you will probably never need to.

*BinHex4*'s .hqx is a standard used to convert Macintosh binary files to an ASCII format primarily for transmission over channels that do not support binary data, like ARPAnet and Usenet mail/news. *StuffIt* allows you to both encode and decode *BinHex4* files.

## Encoding

Select the **Encode BinHex File…** command from the **Other** menu. Select the file to encode. Name the resulting file and select the destination. As a convention, *BinHex4* files usually have the ".hqx" suffix

## Decoding

Decoding works the same way. (Of course, use the **Decode BinHex File…** command instead of encode.)

## Include LFs

On some systems, like my own site, the mailing of .hqx files may not be reliable unless linefeeds are included in addition to carriage returns. If this option is checked, *StuffIt* will include linefeeds when encoding *BinHex4* files.

### Editorial Comment

*BinHex4* encoding is very inefficient, encoding 3 bytes into 4 characters, or a 33% increase plus some other overhead. The UNIX utility btoa encodes 4 bytes into 5 characters, resulting only in a 25% size increase. I guess the author of *BinHex4*, Yves Lempereur, doesn't work with UNIX systems. Rather ironic considering that about the only place *BinHex4* is still used is on UNIX sites on the net. Hey, let's forget it and use *btoa*/*atob*!

### Technical Note

*StuffIt* does not perform RLE encoding on *BinHex4* files. It will decode it, though. RLE is generally not too effective anyway. Thus, *StuffIt* may encode .hqx files to a size slightly larger than *BinHex4* does. However, *BinHex4* can decode the files and*StuffIt* can decode .hqx files with RLE.

# A. Legal Information

I hate fine print. I avoid disclaimers and the like, but there are a few things I might mention. Most of this belongs by the copyright notice, but I don't like it there. So sue me, figuratively of course.

Apple, Macintosh and *MacWrite* are registered trademarks of Apple Computer, Inc.
*Finder* and *MultiFinder* are trademarks of Apple Computer, Inc.

Times is a registered trademark of Linotype Co.

*LightspeedC* is a trademark of THINK Technologies, Inc.

GEnie is a trademark of General Electric Information Services, USA.

CompuServe is a registered trademark of CompuServe, Inc.

MacNET is a registered trademark of Connect, Inc.

Delphi is a trademark of General Videotex Corp.

The names and words *StuffIt*, *UnStuffIt*, *StuffIt Utilities*, sit, .sit and unsit; the black, fading, shadow *StuffIt* logo; and the phrase, "The Macintosh archive utility" should be used to refer to my programs only. They may be used with programs compatible with and/or requiring *StuffIt*, *UnStuffIt* and/or *StuffIt* archives. *StuffIt*, *StuffIt Utilities* and *UnStuffIt* should not be used as titles of programs for any computer. Please use sit and unsit instead. I have devised the above "rules" to prevent general confusion within the computing community. You are not legally bound to follow these rules, but I would prefer it to keep uniformity and not to confuse users.

# B. Error Codes

The more common file system errors are now reported as a verbal message. However, some are still reported in the form of cryptic error codes. The following list should help you indentify these errors. The codes have been grouped logically as to why they may have occurred.

```
DirFulErr     = - 33; { Directory full }
DskFulErr     = - 34; { disk full }
IOErr         = - 36; { I/O error - Possibly bad disk?  }
MFulErr       = - 41; { memory full(open) or file won't fit (load) }
TMFOErr       = - 42; { too many files open }
                      { These are basically self-explanatory. }

WPrErr        = - 44; { disk is write protected }
FLckdErr      = - 45; { file is locked }
VLckdErr      = - 46; { volume is locked }
OpWrErr       = - 49; { file already open with with write permission }
PermErr       = - 54; { permissions error (on file open) }
WrPermErr     = - 61; { write permissions error }

lastDskErr    = - 64; { last in a range of low level disk errors }
firstDskErr   = - 84; { first in a range of low level disk errors }
                      { These errors may indiate defective hardware or a bad disk. }

TMWDOErr      = - 121;{ No free WDCB available }
                        { This error is caused by a limitation in the Mac's file system. }
                        { Try quitting the current application and try again. }
```

Some error messages which may be reported by *StuffIt* and which may be a bit cryptic…

After selecting a file in any of the Open type dialog boxes or after typing in a name in any of the Save As type dialog boxes, you may get a report that a “serious error” has occurred. This should be documented in your Macintosh manuals, but I haven’t personally checked. This error is identical to the TMWDOErr (-121). It is caused by a limitation in the Mac’s file system. Quit back to the *Finder*, relaunch and try again.

# C. UnStuffIt

*UnStuffIt* is a utility which will only extract non-encrypted entries from archives and join segments. It works like *StuffIt* in many ways, so if you're familiar with *StuffIt*, you should know how to operate it.

There is a difference in how the **Comments** icon works. Since *UnStuffIt* cannot change archive comments, if none exist, the icon is dimmed, rather than showing an empty "balloon" as in *StuffIt.*

I do not intend to update *UnStuffIt* much further except for bug fixes. Its purpose is to remain relatively small (43k) so that it can fit onto a floppy easily, thus, making life easier for those of you giving archives to friends. *UnStuffIt* takes up less than 200k of RAM, so fit can fit under *MultiFinder* for those who don't have a lot of memory to spare. *UnStuffIt* is a freeware utility.

**Note:** Although free, like *StuffIt*, distributing *UnStuffIt* with commercial packages and/or demo's requires special licensing.

# D. Technical Information

For those interested in how *StuffIt* stores its archives:

## Cyclic Redundancy Check

For error detection, *StuffIt* uses a standard variation of CRC-16. The variation is the same one as found in many IBM PC programs, and is thus known as PC-CRC-16 in some circles. It is also widely used in the UNIX world, so some just refer to it as plain CRC-16. It is the same CRC-16 that Arc 5.12 uses. Note however that it is **not** CCITT CRC-16. The magic values used are a tad different. (Then again, *PackIt* claims to use CCITT when it uses XMODEM style CRC. I had “fun” figuring that one out.)

```
/* CRC computation logic

   The logic for this method of calculating the CRC 16 bit polynomial
   is taken from an article by David Schwaderer in the April 1985
   issue of PC Tech Journal.

   Not quite CCITT-CRC16... and definitely not xmodem CRC16.
*/

int crctab[] =                          /* CRC lookup table */
{      0x0000, 0xC0C1, 0xC181, 0x0140, 0xC301, 0x03C0, 0x0280, 0xC241,
       0xC601, 0x06C0, 0x0780, 0xC741, 0x0500, 0xC5C1, 0xC481, 0x0440,
       0xCC01, 0x0CC0, 0x0D80, 0xCD41, 0x0F00, 0xCFC1, 0xCE81, 0x0E40,
       0x0A00, 0xCAC1, 0xCB81, 0x0B40, 0xC901, 0x09C0, 0x0880, 0xC841,
       0xD801, 0x18C0, 0x1980, 0xD941, 0x1B00, 0xDBC1, 0xDA81, 0x1A40,
       0x1E00, 0xDEC1, 0xDF81, 0x1F40, 0xDD01, 0x1DC0, 0x1C80, 0xDC41,
       0x1400, 0xD4C1, 0xD581, 0x1540, 0xD701, 0x17C0, 0x1680, 0xD641,
       0xD201, 0x12C0, 0x1380, 0xD341, 0x1100, 0xD1C1, 0xD081, 0x1040,
       0xF001, 0x30C0, 0x3180, 0xF141, 0x3300, 0xF3C1, 0xF281, 0x3240,
       0x3600, 0xF6C1, 0xF781, 0x3740, 0xF501, 0x35C0, 0x3480, 0xF441,
       0x3C00, 0xFCC1, 0xFD81, 0x3D40, 0xFF01, 0x3FC0, 0x3E80, 0xFE41,
       0xFA01, 0x3AC0, 0x3B80, 0xFB41, 0x3900, 0xF9C1, 0xF881, 0x3840,
       0x2800, 0xE8C1, 0xE981, 0x2940, 0xEB01, 0x2BC0, 0x2A80, 0xEA41,
       0xEE01, 0x2EC0, 0x2F80, 0xEF41, 0x2D00, 0xEDC1, 0xEC81, 0x2C40,
       0xE401, 0x24C0, 0x2580, 0xE541, 0x2700, 0xE7C1, 0xE681, 0x2640,
       0x2200, 0xE2C1, 0xE381, 0x2340, 0xE101, 0x21C0, 0x2080, 0xE041,
       0xA001, 0x60C0, 0x6180, 0xA141, 0x6300, 0xA3C1, 0xA281, 0x6240,
       0x6600, 0xA6C1, 0xA781, 0x6740, 0xA501, 0x65C0, 0x6480, 0xA441,
       0x6C00, 0xACC1, 0xAD81, 0x6D40, 0xAF01, 0x6FC0, 0x6E80, 0xAE41,
       0xAA01, 0x6AC0, 0x6B80, 0xAB41, 0x6900, 0xA9C1, 0xA881, 0x6840,
       0x7800, 0xB8C1, 0xB981, 0x7940, 0xBB01, 0x7BC0, 0x7A80, 0xBA41,
       0xBE01, 0x7EC0, 0x7F80, 0xBF41, 0x7D00, 0xBDC1, 0xBC81, 0x7C40,
       0xB401, 0x74C0, 0x7580, 0xB541, 0x7700, 0xB7C1, 0xB681, 0x7640,
       0x7200, 0xB2C1, 0xB381, 0x7340, 0xB101, 0x71C0, 0x7080, 0xB041,
       0x5000, 0x90C1, 0x9181, 0x5140, 0x9301, 0x53C0, 0x5280, 0x9241,
       0x9601, 0x56C0, 0x5780, 0x9741, 0x5500, 0x95C1, 0x9481, 0x5440,
       0x9C01, 0x5CC0, 0x5D80, 0x9D41, 0x5F00, 0x9FC1, 0x9E81, 0x5E40,
       0x5A00, 0x9AC1, 0x9B81, 0x5B40, 0x9901, 0x59C0, 0x5880, 0x9841,
       0x8801, 0x48C0, 0x4980, 0x8941, 0x4B00, 0x8BC1, 0x8A81, 0x4A40,
       0x4E00, 0x8EC1, 0x8F81, 0x4F40, 0x8D01, 0x4DC0, 0x4C80, 0x8C41,
       0x4400, 0x84C1, 0x8581, 0x4540, 0x8701, 0x47C0, 0x4680, 0x8641,
       0x8201, 0x42C0, 0x4380, 0x8341, 0x4100, 0x81C1, 0x8081, 0x4040
};

/* CRC value is initialized to zero (0) at beginning.  Then the following statement is used
```

```
      to compute the new CRC value after each byte:

      crcval = (((crcval>>8)&0x00ff) ^ crctab[(crcval^(c))&0x00ff]);

      …where c is the byte
*/
```

CRC checking is relatively safe. The odds of two different files having the same CRC are very slim. I'm sure you can find numbers in the appropriate text books if you desire them… ("Left as an exercise for the reader"?)

## Archive Format

A *StuffIt* archive has the following format:

```
/* StuffIt.h: contains declarations for SIT headers */

typedef struct                              /* 22 bytes */
{     OSType            signature;          /* = 'SIT!' -- for verification */
      unsigned int      numFiles;           /* number of files in archive */
      unsigned long     arcLength;          /* length of entire archive
                                               incl. hdr. -- for verification */
      OSType            signature2;         /* = 'rLau' -- for verification */
      unsigned char     version;            /* version number (1 for now) */
      char              reserved[7];
} sitHdr;

typedef struct                              /* 112 bytes */
{     unsigned char     compRMethod;        /* rsrc fork compression method */
      unsigned char     compDMethod;        /* data fork compression method */
      unsigned char     fName[64];          /* a STR63 */
      OSType            fType;              /* file type */
      OSType            fCreator;           /* er… */
      int               FndrFlags;          /* copy of Finder flags.  For our
                                               purposes, we can clear:
                                               busy,onDesk */
      unsigned long     creationDate;
      unsigned long     modDate;
      unsigned long     rsrcLength;         /* decompressed lengths */
      unsigned long     dataLength;
      unsigned long     compRLength;        /* compressed lengths */
      unsigned long     compDLength;
      int rsrcCRC;                          /* crc of rsrc fork */
      int dataCRC;                          /* crc of data fork */
      char rPad,dPad;                       /* pad bytes-valid only for encrypted entries */
      char reserved[4];                     /* invalid data for now */
      int hdrCRC;                           /* crc of file header */
} fileHdr;

/* Note that the backup date is not saved or restored */

/* file format is:
               sitArchiveHdr
                        file1Hdr
                                 file1RsrcFork
                                 file1DataFork
                        file2Hdr
                                 file2RsrcFork
                                 file2DataFork
                        .
                        .
                        .
                        fileNHdr
```

```
                                fileNRsrcFork
                                fileNDataFork
*/


/* A new feature with v1.5 and later is support for HMFs.  They look like regular entries for
compatibility… */

      Hiearchy Maintained Folder entry format:

               startFolder header (folder1)
               .
               .        file1 in folder1
               .
               .        startFolder header for subfolder (sub1)
               .        .        files/folders in sub1 (folders will be bracketed by
               .        .                                   startFolder and endFolder)
               .        endFolder header (sub1)
               .
               .        Other files and/or folders in folder1
               .        .
               .        .
               .        .
               .        fileN/subN
               endFolder header (folder1)

               startFolder format:
                        data compression method will be startFolder (32)
                        rsrc compression method will be startFolder (32)
                        If start of top most level folder
                                 Comp. Length will be entire compressed length of folder
                                          contents, embedded startFolders and
                                          all endFolders.  (top most startFolder excluded)
                                 Decompressed length will be entire decompressed size of
                                          folder contents
                                 fName contains folder name
                                 creationDate and modDate are valid
                                 Other fields may be invalid
                        otherwise
                                 fName contains folder name
                                 All other fields invalid

               endFolder format:
                        Data compression method will be endFolder (33)
                        Other fields may be invalid


               To maintain compatibility, the header structure used is the same as
                        fileHdr.  This may prove to be quite wasteful, but that's life!


/* A new feature of StuffIt starting with v1.40 is encryption.  Encrypted entries look
like regular entries for compatibility, but if the comp?Method shows encryption, the
following format is used.

               .
               .
               .
      File(N-1)Entry (if any)

      FileNHdr (non-encrypted)

               Encrypted fileNRsrcFork (padded to next 8 bytes (if length % 8 is not 0) )
               Encrypted fileNRsrcFork Key (8 bytes)
```

```
            Encrypted fileNDRsrcFork IV (8 bytes)

            Encrypted fileNDataFork (padded to next 8 bytes (if length % 8 is not 0) )
            Encrypted fileNDataFork Key (8 bytes)
            Encrypted fileNDataFork IV (8 bytes)

      File(N+1)Entry (if any)
            .
            .
            .
```

Details to the NewDE encryption algorithm may be obtained from Richard Outerbridge. (GEnie: Outer; Usenet: outer@utcsri) Public domain C source code is available. He can either provide it or point you in the right direction. Please don't bother the busy man if you don't really need it. Basically, it's just DES w/o the initial and inverse permutations.

*StuffIt*'s implementation of NewDE…

Passwords are resolved into a 64 bit key (master archive key). The key schedule information is available from Rich. A master archive seal is calculated for the key. This seal is a combination of a check value and some random data. Again, ask Rich for details. This seal is stored with any archive that has a password entered as a resource of type 'MKey', ID 0. This seal is required to decrypt entries. A damaged or missing seal is bad news.

For each encrypted fork, a new file key and initialization vector are needed. Versions of these, encrypted with the master archive key, are stored along with the fork. The proper master archive key and seal are needed to decrypt these. Once decrypted, the key is used to decrypt the rest of the encrypted fork. The IV is xor'ed with the first block (8 bytes) of the fork. The first block is then encrypted. The encrypted block is xor'ed with the next block, which is then encrypted and xor'ed with the block after that, etc. More details on how to decrypt file keys given the master archive key and seal are available from Rich.

Encryption is performed **after** the entry is added to the archive. (i.e.: It's performed on the compressed fork if compression was used.)

That's about it on encryption. Here are some constants you may need to know:

```
/* compression methods */
#define noComp 0        /* just read each byte and write it to archive */
#define rleComp 1       /* RLE compression */
#define lzwComp 2       /* LZW compression (was lpzComp in previous documentation) */
#define hufComp 3       /* Huffman compression */

#define encrypted 16    /* bit set if encrypted.  ex: encrypted+lpzComp */

#define startFolder 32  /* marks start of a new folder */
#define endFolder 33    /* marks end of the last folder "started" */

/* all other numbers are reserved */
```

Archive comments are stored as a pure text resource of type SitC and ID 0 with the archive. The length of the comment is the length of the resource.

Archives are of creator 'SIT!' and type 'SIT!'.

Segments are of creator 'SIT!' and type 'SegM'. They have the following format.

```
typedef struct
{     int       magicNumber;/* a hopefully unique id tag.  Generated randomly.  All segments
                              of the same file should have the same magicNumber. */
      int       part;       /* 1..n for segment # */
      char      fName[64];
      long      fType;
```

```
        long      fCreator;
        int       fdFlags;
        long      creationDate;
        long      modDate;
        long      rsrcLength;
        long      dataLength;
        int       hdrCRC;  /* CRC of all bytes up to, but excluding, this one */
        int       rsrcCRC; /* invalid except for the ones stored in the final segment. */
        int       dataCRC;
} segmentHdr;     /* 100 bytes? I'm not sure...  Just did a quickie in my head. */

/* format:
        seg 1:
                  hdr1
                  contents1
        seg 2:
                  hdr2
                  contents2
        ...
        seg N:
                  hdrN
                  contentsN

                  contents is contiguous stream of:
                            rsrc fork
                            data fork

        hdr of seg N (last segment) contains valid fork CRCs
*/
```

I will not attempt to describe the compression algorithms used by *StuffIt*. They've been described elsewhere in a manner clearer than I could ever describe them. I will just refer you to those places.

RLE used by *StuffIt* is the same algorithm as that used by Arc 5.12. Consult the Arc 5.12 sources which should be readily available.

LZW used by *StuffIt* is similar to that used by Arc 5.12 but with a few values changed. *StuffIt* implements LZW at 14 bits whereas Arc 5.12 calls 12 bits crunching and 13 bits squashing. You can use the source provided in the ArcSquash.c file if you change the bits to 14 and the hash table size to 18013. Both of these are in the #defines. Alternatively, if you consult the Compress sources, which are in the public domain and also readily available, you can use the code there. It's almost identical to the Arc 5.12 Squash code except that it contains a bunch of preprocessor macros to define the number of bits and hash table size. Well, I'm sure you can figure out how to set the bits to 14 and the hsize to 18013…

Huffman used by *StuffIt* is the same algorithm as used by *PackIt II* except that it's performed on each fork individually. Consult the *PackIt II File Format* documentation for more information.

If you want to read about compression:

1. T. Welch, "A Technique for High-Performance Data Compression", *IEEE Computer*, Vol. 17, No. 6, June 1984, pp. 8-18.

2. J. Ziv and A. Lempel, "A Universal Algorithm for Sequential Data Compression", *IEEE Trans. Information Theory*, Vol. IT-23, No. 3, May 1977, pp. 337-343.

3. J. Ziv and A. Lempel, "Compression of Individual Sequences via Variable-Rate Coding", *IEEE Trans. Information Theory*, Vol. IT-24, No. 5, Sept. 1978, pp. 5306.

Any good algorithms book should mention enough about the Huffman technique for you to figure out what's going on.

Oh, don't bother with an Arithmetic Coding as described in a *CACM* article a while ago. I've implemented it and it doesn't work too well. It rarely beats Huffman (only surpasses it for large files) and when it does, only by about 1K. This is an empirical observation based on trying an implementation of the algorithm on a variety of Mac files. It's also quite slow.

Interested in cryptology? Here are some references that Rich mentioned to me:

1. "Announcing the Data Encryption Standard and Specifications for the Data Encryption Standard", *Federal Information Processing Standards Publication 46*, U.S. Department of Commerce, National Bureau of Standards, National Technical Information Service, Jan 15, 1977.

[and of course]

2. R. Outerbridge, "Some Design Criteria for Feistel-Cipher Key Schedules", *Cryptologia,* Vol. 10, No. 3, July 1986, pp. 142-156.