MacNetTools

Version 1.2.1

MacNetTools

Version 1.2.1

Introduction

Essential Info

Information of particular importance in this documentation are bracketed with red Essential Info and End Essential Info markings. Even if you read no other parts of this document, at least give yourself the advantage of skimming through it all and reading those parts tagged as essential.

In all of the descriptions that follow, the abbreviation “MNT” will often be used for MacNetTools.

End Essential Info

MacNetTools - what it is and what it does

MacNetTools is a small (but rapidly growing) program for Macintosh which provides a collection of utilities which are commonly needed to perform auxiliary internet-related functions plus a number of other functions related to files and desktop management. The following is an abbreviated list of the functions currently provided. Many of these commands have a wide variety of options and output formats which are fully explained in the later section describing each command. These commands are:

Joins Segmented Files within a folder

Splits Files into folders of segments

Decodes Encoded Binary Files (for each format show below)

- Auto Detect Format
- UUEncoded
- Mime (including AppleSingle and AppleDouble)
- Binhex
- Macbinary
- yEnc

Encodes Binary Files (for each format show below)

- UUEncoded
- Mime
- Mime AppleSingle
- Mime AppleDouble
- Scripted Mime (Future addition)
- Binhex
- Macbinary
- yEnc

List Files

All Files
Uncommented Files
Missing SIT Segments
Missing MPG Segments
Missing Pattern Segments
Pattern
Duplicate Files (based on content, not name)
Set Match
List Match
List Invisibles

Wide variety of output formats and targets

Set Filetype

Sets the Finder filetype of all files dropped on it.

Adjust All Filetype

Sets the Finder filetype of all files dropped on it based on each file's extension.

Concatenate

A very generalized version of the Join command.

Input as:
- Files
- Text Clippings

Combined Output as:
- File
- Text Clipping
- To Clipboard

CRC and Checksums

Sum32
Adler32
XOR8
MacBinary
BinHex
CRC16
CCITT16A
CCITT16B
CRC32
CCITT32
POSIX.2
ZIP 32
MD5
SHA
Custom CRCs
Custom CRC Table Generation

Several output formats and targets

File Tools

Fix HL Files
Strip Resource Fork
Encrypt-Decrypt
Move Matching Files
Move Non Matching Files
Trash Matching Files
Delete Matching Files
Label Matching Files
Rename Matching Files
Rename By Subject
Fix Filenames
Move Duplicates (based on content, not name)
Move Dupl Names (based on name)
DTList To DBF
Alias To File
Dup.nn To Dupa.ext
Fix BinHex

Folder Tools

Combine Fldr
Group Fldr A
Group Fldr B
Group Fldr C
Step Fndr Fldrs
Delete Empty Fldrs
Open Fldrs
Close Fldrs
Calc Fldr Size - not yet functional

Miscellaneous Utilities

Get/Set File Info
Wildcard Match Check
Step Fndr Fldrs
Open Fldrs In Trash
Close Fldrs In Trash
SlideShow (Possible Future addition)

Most of the above functions can be performed either via menu command or via a drag and drop operation onto the program's icon or alias. Most of these functions are performed in a multi-threaded environment to allow background processing which does not interfere with or significantly slow down other tasks which are running on your system at the same time. MacNetTools can be set to run with any number of threads other than the main thread simply with the change of one program variable. Some functions, however, require a distinct order of processing which could easily be upset by use of multiple threads. Consequently, the program is limited to run with only one additional thread over the main thread. The system requirements are as follows:

AppleEvents are required (late System 6 or early System 7)
Thread Manager is required
Appearance Manager is required
CarbonLib is required for Carbon version

All are available in MacOS 8.1 for sure and some perhaps as early as 7.5

Getting Started

Installation

Installation of MacNetTools is simple. Just drag the MacNetTools folder to wherever you want to put it on your hard disk. The first time that you start the program, you will be asked to read and agree to the terms of the license agreement. Other than that there are no further installation steps or copy protection measures. Version 1.0 of MNT included a weak password ("Duck") that had to be entered to force you to read the "ReadMe" file where that password was located (and we all know how Mac users just hate to have to read any documentation), but even this password is gone now. You are on your own and if you choose not to read the provided information, then that is your own weakness.

MacNetTools Modes

Unlike previous versions of MacNetTools, version 1.2 no longer depends on the Drop menu to determine what operation to perform. This was one of the main weaknesses of previous versions and one of the biggest improvements provided in this revision. Previously it was necessary to start up MNT, select the desired operation from the Drop menu and select any needed options in the Prefs dialog. Only then was it possible to drop the desired files and folders onto MNT for processing. This has all changed now and the entire process has been turned into one smooth operation regardless of the function to be performed and regardless of the current settings in the program. However, I have not taken anything away from you in the process. If there happen to be any users out there who really did like the way that MNT used to work (although I can't possibly imagine why), you can still use MNT by the old method.

MNT now has 4 modes of operation with each of those being applicable to two different sources of input data. For those of you who cannot or prefer not to understand these modes, all is not lost because when it is first run, MNT sets its default parameters to those which you are most likely to want to use. However, if you are like most Mac users, chances are very good that you will start off by simply fiddling around with everything in sight just to see what it does. Since the effects of all MNT settings are delayed until the function is actually applied to the files/folders in question, there is a very good chance that such experimentation will get you into a "no man's land" program state where what you want to do is unlikely to be what you get (even my beta tester did this). If you absolutely feel that you must do such experimentation rather than reading this documentation that tells you how it really is supposed to work, then to give yourself a fighting chance of getting things right, at least trash the "MacNetTools Prefs" file in the Preferences folder of your active startup disk whenever you get serious about trying to use MNT the proper way.

Essential Info

MNT receives its input info describing which files/folders it is to operate upon from two sources:

1) User dropping of files/folders onto MNT icon (User Drop Input)
2) Commands sent to MNT by other programs (Helper Program Input)

For each of these input sources, the command which MNT executes is determined in one of 4 ways.

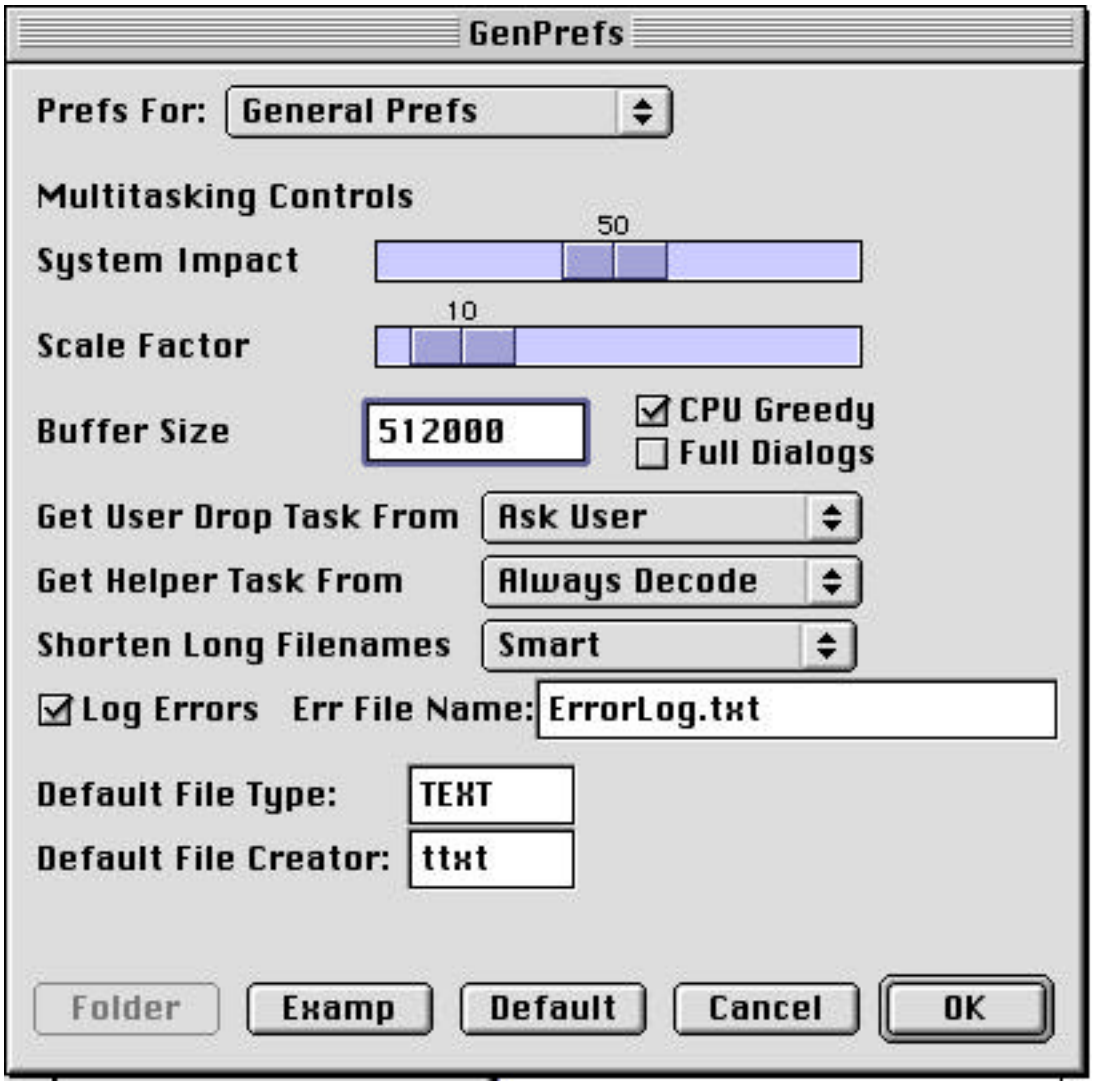
1) Get Command from Drop Menu
2) Ask User to Select Command from List (Default case for UserDrop, N/A for Helper mode)
3) Always Execute Decode Command (Default case for Helper mode)
4) Repeat Last Command that was Executed

Unless you make any changes to the default settings (see paragraph above), the following rules will apply. MNT will always execute the Decode command when acting as a Helper application. MNT will always ask the user to select the command to perform when used to handle items which are dropped onto

the MNT program icon or its alias. Almost without exception, the only use for Helper applications is to decode binary attachments, so this is the appropriate choice for the Helper mode. In the unlikely event that another operation is needed in Helper mode, the user can always select “Drop Menu” for that source and set the drop menu to the desired operation. Helper applications are becoming a thing of the past though as more and more newsreaders and web browsers include their own decoders, so it is a use that you may not encounter very often. If you do need such a helper application, you can simply select MNT as that application and set up the Helper setting in MNT appropriately. On the other hand, using MNT to batch process raw downloaded files is a very efficient way to deal with large amounts of data in a very short time. This technique will be fully described later in this documentation.

End Essential Info

The following series of pictures shows the General Prefs pane of the preferences dialog which is where the parameters described above are set. The method of command determination that takes place when the user drops files or folders onto MNT is selected by the popup menu labeled “Get User Drop Task From”. The method of command determination that takes place when another program uses MNT as a Helper app is selected by the popup menu labeled “Get Helper Task From”. Unless you really understand what you are doing it is strongly suggested that you do not change these two settings.

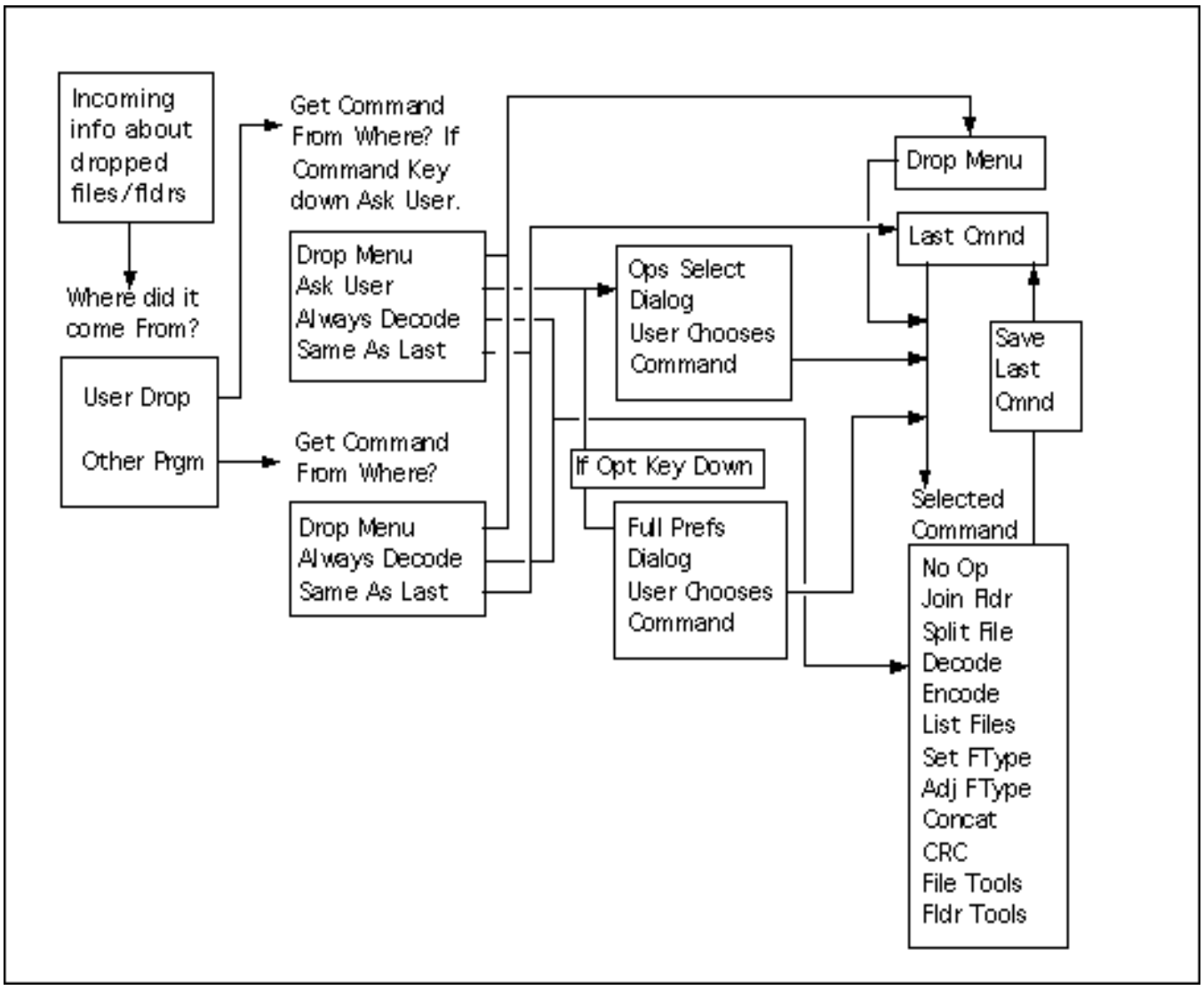


The following picture fragment shows the “Get User Drop Task From” menu pulled down.

The following picture fragment shows the “Get Helper Task From” menu pulled down.

MacNetTools Command Flow

The following drawing shows an overall block diagram of the processing of incoming events received by MNT. It is rather difficult to follow (even for me) and I don't really expect anyone to be able to understand it any better, but here it is in case anyone should want to see what is going on at a glance. Perhaps I can redraw this better for future documentation rewrites. In the meantime, the written description of what happens follows this drawing and should be a lot easier to understand.



How MNT Processes Files That Are Sent To It

Essential Info

MNT is invoked whenever another application sends files to it for processing or whenever the user drops files or folders onto the MNT program icon or its alias for processing. The first thing that happens is that MNT determines who sent the files or folders to it.

1) If the files/folders were sent to MNT by another program, then MNT looks at the "Get Helper Task From" setting.

   A) If Get Helper Task is Drop Menu, then MNT executes the command currently selected on the Drop Menu.

   B) If Get Helper Task is Always Decode, then MNT executes the Decode command with these files/folders

   C) If Get Helper Task is Same As Last, then MNT repeats the last command with these files/folders

2) If the files/folders were sent to MNT by a user drop, then MNT looks at the "Get User Drop Task From" setting. If the Command Key is held down at the time of this user drop, then the "Get User Drop Task From" setting is ignored and MNT always asks the user to select the command (item B below).

   A) If Get Helper Task is Drop Menu, then MNT executes the command currently selected on the Drop Menu.

   B) If Get Helper Task is Ask User, then MNT presents the Ops Select dialog for the user to select the command to perform. If the Option Key is held down, then MNT presents the Full Prefs dialog for the user to select the command to perform. The Full Prefs has ALL of the options available to choose from for each command, whereas the Ops Select dialog has only a subset of those most frequently needed options for each command in order to avoid confusion by the many many options which are available with some of the commands. Otherwise the two sets of dialogs are identical. If the OK button is clicked in either of these two dialogs, then the command is executed. If the Cancel button is clicked, then nothing happens.

   C) If Get Helper Task is Always Decode, then MNT executes the Decode command with these files/folders

   D) If Get Helper Task is Same As Last, then MNT repeats the last command with these files/folders

3) If a command is executed, that command is remembered as the last executed command for future use by the "Same As Last" option.

4) If a selection is made on the Drop menu, that selection is remembered as the Drop Menu item for future use by the "Drop Menu" option.

5) All settings are remembered between program runs.

Under typical usage, the user will drop files or folders (as appropriate to the command) onto the MNT program icon or alias. The program will start up (if it is not already running) and present the user with the Ops Select dialog (or Full Prefs dialog if the Option Key is held down) for the user to select a command for MNT to execute. The user will then select the desired command from the Task/Prefs popup menu (see below) which will cause the Prefs panel for that command to be displayed. The user then selects any

special options which are desired and clicks the OK button which will cause that command to be executed. If the user changes his mind, he can always click the Cancel button and nothing will happen.

Multiple commands can be given to MNT at once (by dropping multiple files/folders) and additional commands (even mixed commands) can be added to the queue (by dropping yet more files/folders) while MNT is processing the previous batch. Each command will then be queued for processing when the previous batch is done. Somebody is bound to ask "what is a queue" so I will explain now. It is a computer term borrowed from the Queen's English which simply means a line of items waiting for something. The Brits speak of standing in the queue to get show tickets (or whatever). Computer data can get queued while awaiting processing as in the case of MNT. It could also be described as a pecking order.

The settings selected for a group drop of multiple files/folders are applied to all items in that group. However, MNT does not queue independent sets of Prefs data (this might become a future addition). There is only one set of preferences data which gets evaluated at the time a queued task becomes active. If you select particular settings for one set of dropped items, then repeat for a second group using the same settings, then drop a third group while changing the Prefs settings before the second group has become active, then the second group will also end up using the new settings intended to be applied only to the third group. It is very unlikely that this will ever become a problem for anyone though unless your hand is extremely fast and you are dealing with really huge amounts of data. Typically I have found that MNT is so fast that the user cannot easily keep up with it in the commands that are fed to the program.

Most (but not all) commands know about the standard Mac command period abort technique. While executing any command, simply hold down the command key and type a period '.' to abort execution of that command. This command period detection is done outside of the usual Mac event mechanism for sake of speed so you may have to hold down the command period combination for a second or until it is detected.

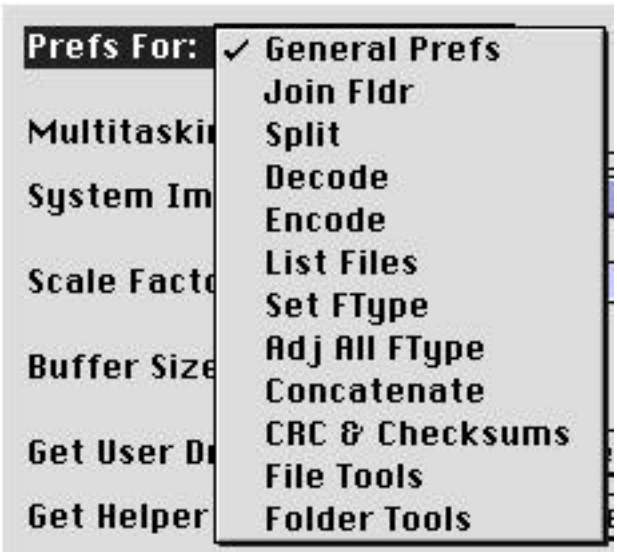
When all commands are completed, MNT will quit if it was not already running when invoked or will just return to the idle state if it was previously running.

End Essential Info

Task/Prefs Popup Menu

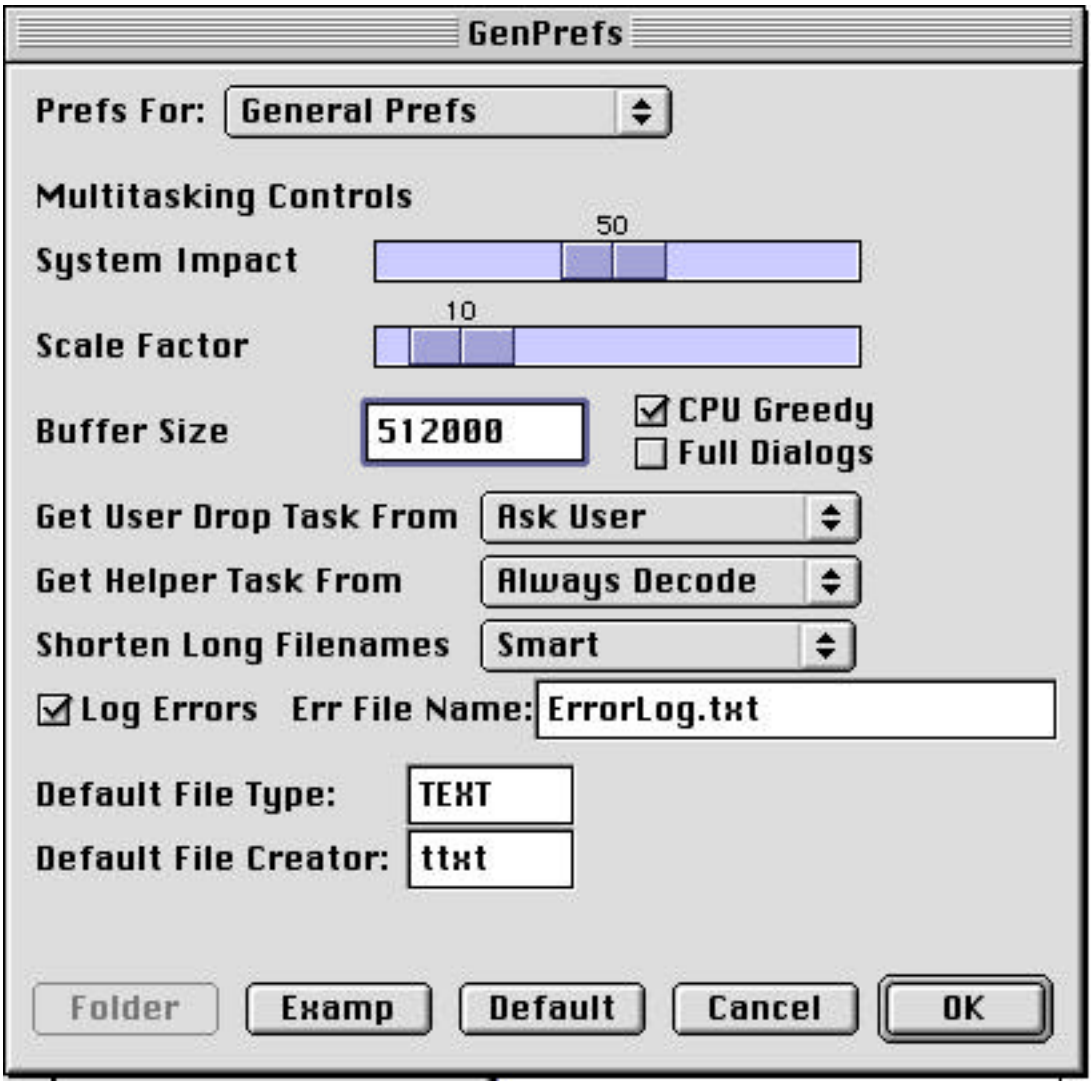
(In pulled down state to show selectable commands)

This popup menu is where you select the command to be executed. It is identical in both the Ops Select dialog and in the Prefs dialog although labeled differently in each. In the Ops Select dialog, it is labeled “Task”. In the Full Prefs dialog, it is labeled “Prefs For”.



The Prefs Dialog

The General Prefs panel of the Prefs dialog is shown below. It is where all of the preferences are set that affect MNT on an application-wide basis. Note that the previous version of MNT had only one Prefs panel. The current version has 12 different panels which are selected for display using the “Prefs For” popup menu. The Preferences dialog serves two purposes. It can be displayed by choosing “App Preferences” from the Edit menu and used to set up any of the 12 desired Prefs panel settings independently of the execution of any command. It can also be displayed by holding down the Option Key while invoking MNT with a file/folder drop in which case the selected settings will be both applied to the currently dropped items and will also be retained for future use (but only if the OK button was pressed or if another panel is selected for display).



Buffer Size - All of the MacNetTools utilities make use of buffered file I/O. This means that the program reads from and writes to disk files in relatively large chunks rather than just a few bytes at a time. Then the program accesses the data in this buffer at its leisure. By avoiding frequent smaller reads and writes, much faster performance is achieved plus the program presents very little interference with other programs which are trying to do disk reads and writes at the same time. For example, when I wrote the new yEnc decoder, it was originally done without buffering while perfecting the code. I forgot to add the buffer code and later became displeased at how slowly it ran. When I saw what I had done and added buffering, the speed improved by a factor of 4. Additionally,

the UU and Mime encode and decode routines use additional buffering techniques designed to enhance their isolation from the rest of the system and thus to speed their performance even more. Having this type of scheme should allow the program to take the greatest advantage of program and data caches on current processors so that a significantly greater number of data accesses occur with cached data which can be accessed several times faster than data in main memory even. The bottom line is that the Buffer Size parameter in the dialog sets up the size of the larger of the two buffers (the size of the smaller buffers is fixed). The default value is 512000 bytes but if you have a lot of memory to burn, you can set this value to several megabytes or more (performance will cease to increase if you set it higher than the systems disk cache size as set in the Memory Control Panel). Just remember that the program must allocate this much memory so if you increase this value significantly, be sure to also increase the application memory (Preferred Size) in the application's Get Info box by six times that amount (two buffers - one for input - one for output, 3 sets of these buffer pairs for decode, recursive decode if enabled and first failed recursive decode).

System Impact - The System Impact slider controls the amount by which the program impacts the rest of the system. It does this by varying the amount of processing which it does before relinquishing control to the system. Faster processors may be so fast that the System Impact parameter may have little effect on the overall system. For this reason the Scale Factor is provided.

Scale Factor - The Scale Factor is simply a factor by which the System Impact setting is multiplied. The product of the two represents the amount of work that is done (typically the number of lines processed) before control is returned to the rest of the system. Since the two numbers are simply multiplied together, either slider could in reality be used for either purpose, however it simplifies matters if one is used for the Impact setting and the other is used to match MNT to the speed of the processor on which it is running. For a G4/800 the 50/10 settings shown in the pics are about right. the original version of MNT was developed on a PM9600 for which a Scale Factor setting of 1 or 2 was appropriate. To find a value that is appropriate to your system, proceed as follows. Find a file to encode that is in the 10-20 MB range (more if your system is exceedingly fast) to give an encoding time of 10 seconds or more. Now open a word processor, text editor (SimpleText will do OK) or any program that will visibly slow down as time is taken away from it. Even copying a large file in the Finder or switching between programs on the Application Menu may show the effect. SimpleText is a good choice because it is easy to see any delay between typing and the appearance of the typed characters in its window. Start MacNetTools, select the General Prefs dialog. Set the System Impact slider to the 100 and the Scale Factor slider to 10 then click OK to accept the changes. Then begin the encode of the large file of your choice. Switch to your other application, begin typing if it is a text program and take note of any sluggishness which results from the encode process. If there is none or very little, increase the value of the Scale Factor, click OK and repeat the process until the amount of sluggishness that is introduced is the most that you are willing to accept. Then return the System Impact slider to a central position and you will have full control over MacNetTools performance using the System Impact slider alone. If you are unwilling or unable to perform this test, then just leave the values set to their default settings. You may not get the maximum performance from MacNetTools, but it will be plenty fast enough to suit most users.

CPU Greedy - this was an attempt to give the user additional control over how much time was given to other executing tasks. As implemented it slowed things down too much when unchecked, so just leave this one checked all the time. I am now thinking about using a new time based (rather than work based) mechanism to control yields to the system in the future releases of MNT. This should be more efficient and work more consistently than the current scheme. When implemented, both the CPU Greedy checkbox and the Scale Factor slider will be removed and the user will be left with one simple control to determine the maximum amount of time which is allowed to pass between yields.

Full Dialogs - This shortcut causes the full Prefs dialog to always be presented to the user in Ask User mode rather than the Ops Select dialog as usual. It works like the user is always holding the Option Key down when dropping items onto MNT. Most newbies will want the Ops Select dialog to avoid becoming confused by the myriad of options which MNT has to offer. However once a user becomes accustomed to all of those options, it can at times be more convenient to always have them available for choice rather than having to take and additional action to access them. The Full Dialogs checkbox makes this possible.

Get User Drop Task From - explained above under “How MNT Processes Files”

Get Helper Task From - explained above under “How MNT Processes Files”

Shorten Long Filenames - this chooses the method by which long Wintel filenames are shortened to a length of 31 characters or less as required under Mac OS versions other than OS X. This choice is used in many places throughout the program by many commands. There are five choices:

Don't Shorten	Use Not advised
At Beginning	Characters are removed at the beginning of the filename
In Middle	Characters are removed in the middle of the filename
At End	Characters are removed at the end of the filename
Smart	Characters are removed in the middle of the filename but the break point at which characters are removed is first adjusted forwards or backwards by up to one eighth the length of the filename so as to remove characters at natural text boundaries which may exist within the filename. This is the default setting.

Log Errors - If checked, MNT will maintain an error log located within the MNT program folder which reports errors incurred by the program, primarily useful for decode operations. A sample log entry is shown below. Since these log entries accumulate indefinitely which would result in increasingly slower performance by MNT, the log is automatically cleared anytime that new errors are logged and the last error that was logged occurred more than an hour ago.

```
Tue, Jul 30, 2002, 15:38:51
File:  MickeyMouse.mpg.000
Error: -5012
Meaning: Item Not Found
```

Err File Name - the user’s choice of filename to be used for the error log... just type the name in.

Default File Type - this is the default filetype used for all output. MNT has other facilities for autotyping of most files as they are created and although this default filetype is available program-wide, it is typically only used as the filetype for text output. Therefore it should be the filetype of a text type file or ‘TEXT’ as shown.

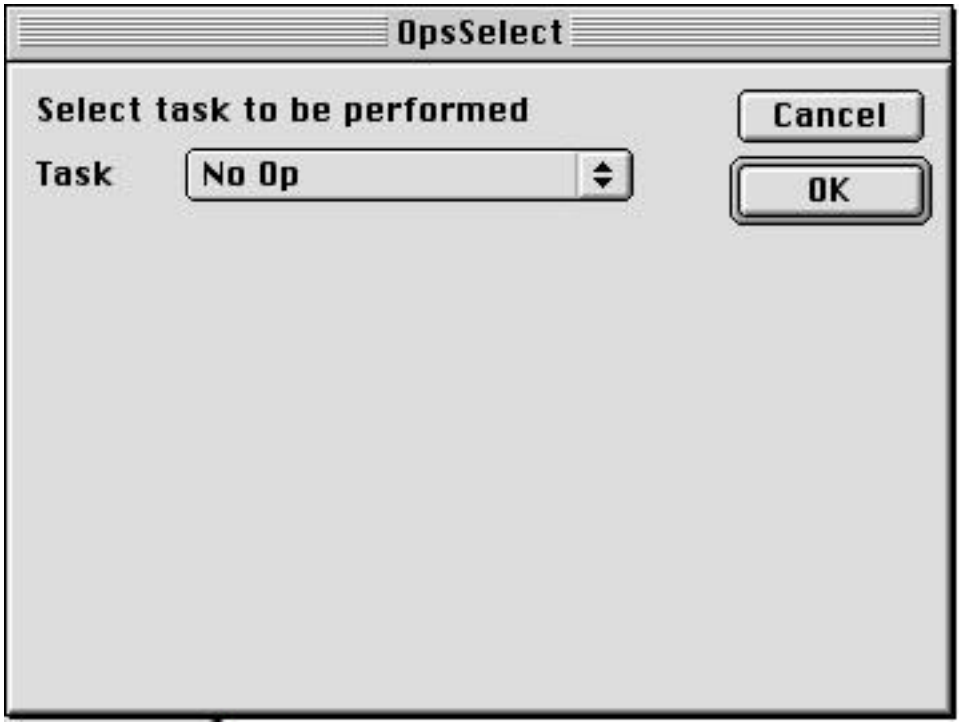
Default File Creator - this is the default file creator used for all output. MNT has other facilities for autotyping of most files as they are created and although this default file creator is available program-wide, it is typically only used as the file creator for text output. Therefore it should be the filetype of any text type file program. The default setting of ‘ttxt’ as shown is the creator code for SimpleText files.

Example Button - brings up a Standard File / Navigation dialog allowing the user to browse for and select an example file of the type and creator desired to be used for the default file Type and Creator. if the user selects Open in that dialog, that file’s Type and Creator are determined and loaded into the Default File Type and Default File Creator fields described above.

Defaults Button - resets all parameters in this panel to their original default values.

The Ops Select Dialog

The Ops Select dialog is the dialog normally seen by the user when dropping files/folders onto MNT (unless the Option Key was also held down at the time of the drop, which as described before would cause the Full Prefs dialog to be displayed). This dialog is also where the user selects the command to be performed via use of the task popup menu. The No Op panel of the Ops Select dialog shown below is the Ops Select complement to the App Prefs panel in the Full Prefs dialog. If executed, the No Op command will cycle through the dropped files/folders but does nothing.



Available Commands

No Op

The No Op command does nothing. Its dialog is the Ops Select complement to the General Prefs panel in the Full Prefs dialog and is shown in the section above.

Join Fldr

The Join Folder command is probably what 90% of you use MNT for. It takes a folder of binary movie segments (or any other split binary files) and joins the parts to recreate the original combined movie. This command normally assumes the use of a common filenaming protocol that is typically used in the multimedia and other large binaries newsgroups whereby file segments have three digit incrementing suffixes which begin with .001. For example we may have an mpeg movie named PopeyeSavesOlive.mpg which has been split into file segments named as follows:

PopeyeSavesOlive.mpg.000
PopeyeSavesOlive.mpg.001
PopeyeSavesOlive.mpg.002
PopeyeSavesOlive.mpg.003
...
PopeyeSavesOlive.mpg.122
PopeyeSavesOlive.mpg.123

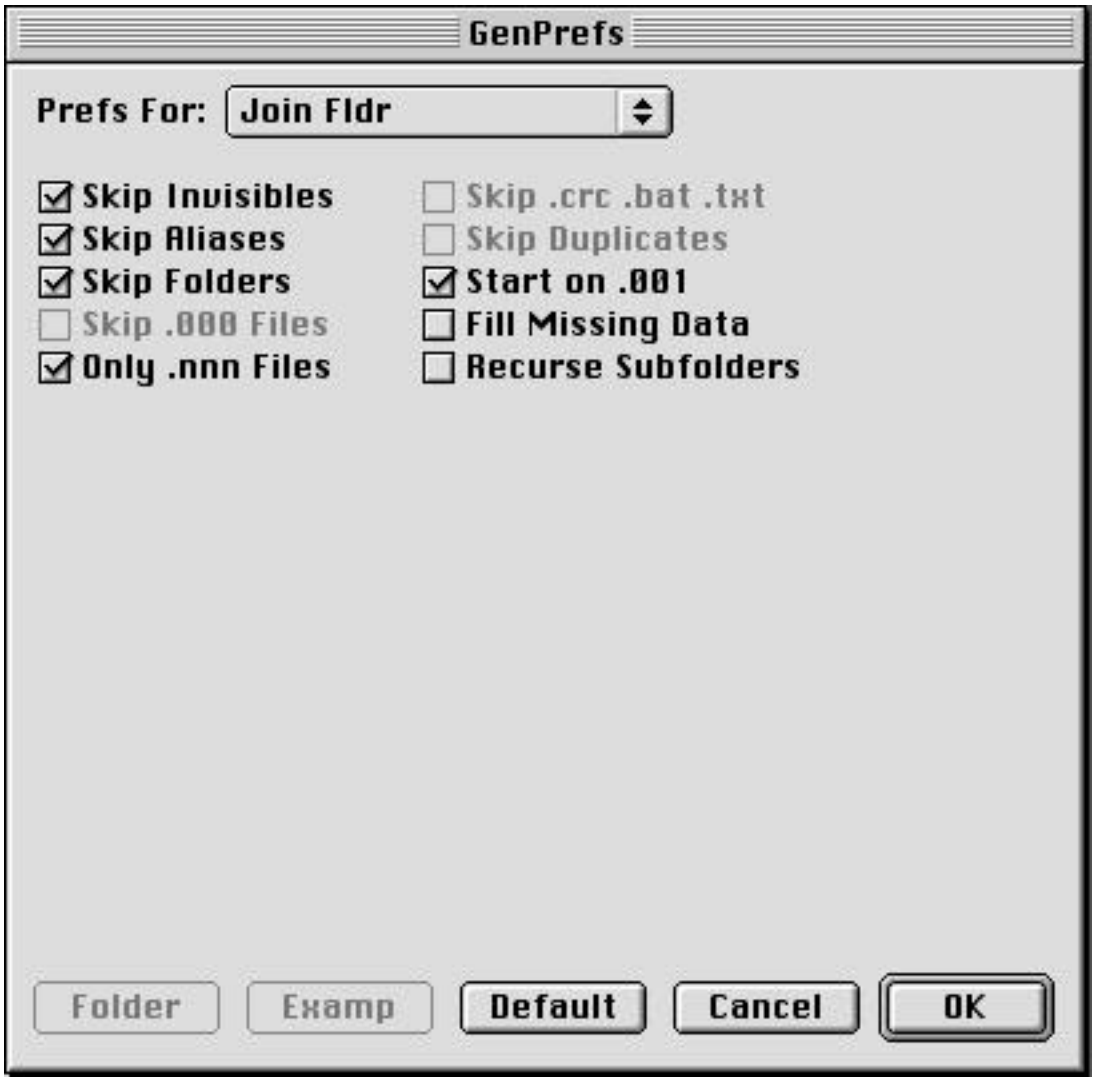
File segments having suffixes of .000 are normally filetype info for Wintel machines and are not used on the Mac. File segments having suffixes of .001 are special (mpeg only) and must be included for an mpeg movie to play since they contain the basic info about the movie in general (movie width, height, color depth, compression type etc). In fact, mpeg .001 files can be played by themselves to give a quick preview of any movie's content.

When the folder containing these file segments is dropped onto MNT and the Join command selected, the file segments will be reassembled in the same folder as the folder containing the dropped folder (i.e. its parent folder) to create a combined movie file with a filename equal to the first segment name with the .nnn removed (or PopeyeSavesOlive.mpg in this case). it will be given a filetype based on the extension of that newly recreated file. In the case of the Join commands, since some Wintel users insist on using an underbar ('_') character in front of the extension, either a period of underbar character will be considered to be a valid first character of the extension.

The original version of MNT provided no control over the Join command. Several features have now been added to give the user better control over the Join process and thus allowing it to be used to Join a wider variety of files. None of these are controls which would frequently be changed from one Join to the next so the Ops Select dialog for Join is empty as shown below and these controls are provided only on the corresponding Full Prefs dialog as shown in the second picture below. Description of these controls follows that picture.

Ops Select dialog for Join Fldr

Full Prefs dialog for Join Fldr



Skip Invisibles - the Macintosh file system has a feature whereby any file can be made invisible to the user. The file system (and MNT's file scanning mechanism) will still see these files and will try to combine them with the movie being joined. A very common example is the invisible file named "Icon" inside of any folder which has ever had a custom icon assigned to it. In this case, that invisible file will precede the .001 file's data and will flat ruin the movie which you are trying to Join. Checking the Skip Invisibles box will cause all invisible files to be skipped by the join process. It should normally be checked.

Skip Aliases - causes MNT to skip all aliases which it encounters in the folder of files to be joined.

Skip Folders - causes MNT to skip all folders which it encounters in the folder of files to be joined.

Skip .000 Files - causes MNT to skip all .000 which it encounters in the folder of files to be joined. This is disabled and no longer needed with the "Start on .001" option which gives the same effect.

Only .nnn Files - causes MNT to join only those files ending in a .nnn where n is any digit from 0-9. This option makes both the "Skip .crc .bat .txt" and "Skip Duplicates" unneeded.

Skip .crc .bat .txt - causes MNT to skip all files ending in ".crc", ".bat" or ".txt" which it encounters in the folder of files to be joined. This is disabled and no longer needed with the "Only .nnn Files" option since a file cannot end in one of the three strings listed and ALSO end in 3 digits.

Skip Duplicates - causes MNT to skip all duplicate files which it encounters in the folder of files to be joined. Duplicates are normally tagged with a ".nn" (where n is any digit from 0-9) following the filename as shown below. The "Only .nnn Files" option excludes those files since they cannot also match the specified .nnn pattern so the Skip Duplicates option is disabled and no longer needed.

Start on .001 - causes MNT to wait until it has detected the .001 segment before it begins to write to the output file. This ensures that the movie will begin with the true beginning segment.

Fill Missing Data - causes MNT to fill in missing spans of data with zeros, both for short and missing segments. Works for all segments except the .001 segment and the last segment. Option included by popular demand.

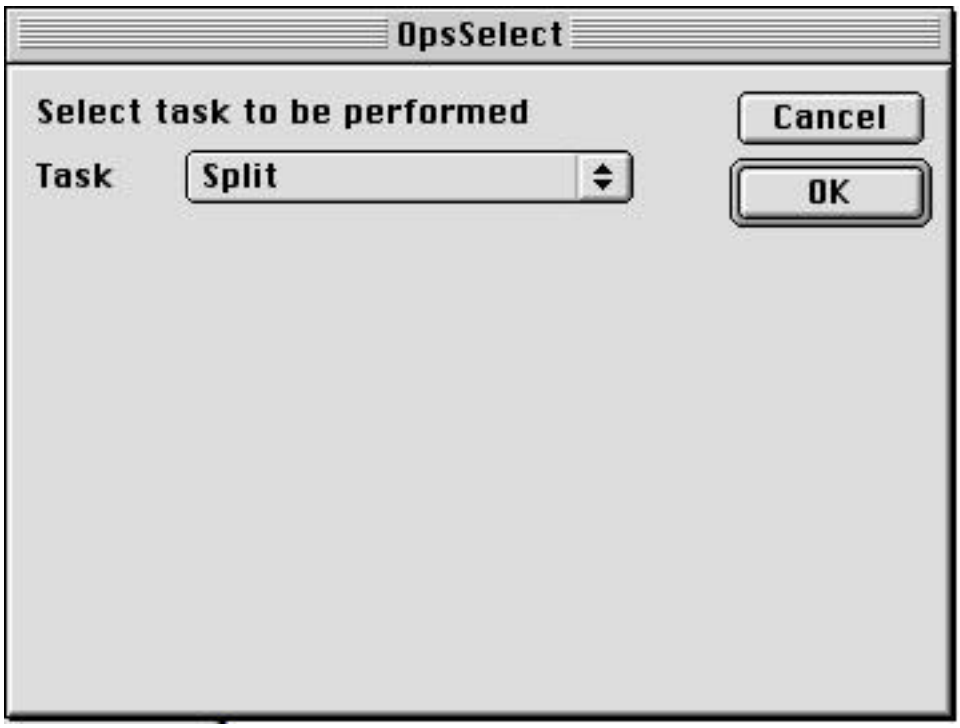
Recurse Subfolders - causes MNT to join the contents of both the main folder and also subfolders within that folder to create the combined output file. This should normally remain unchecked unless you have organized the segments into a very well known order and know the order in which MNT will encounter them (you can determine this with the "List Files" command). Just don't do it unless you are sure.

Defaults Button - resets all parameters in this panel to their original default values.

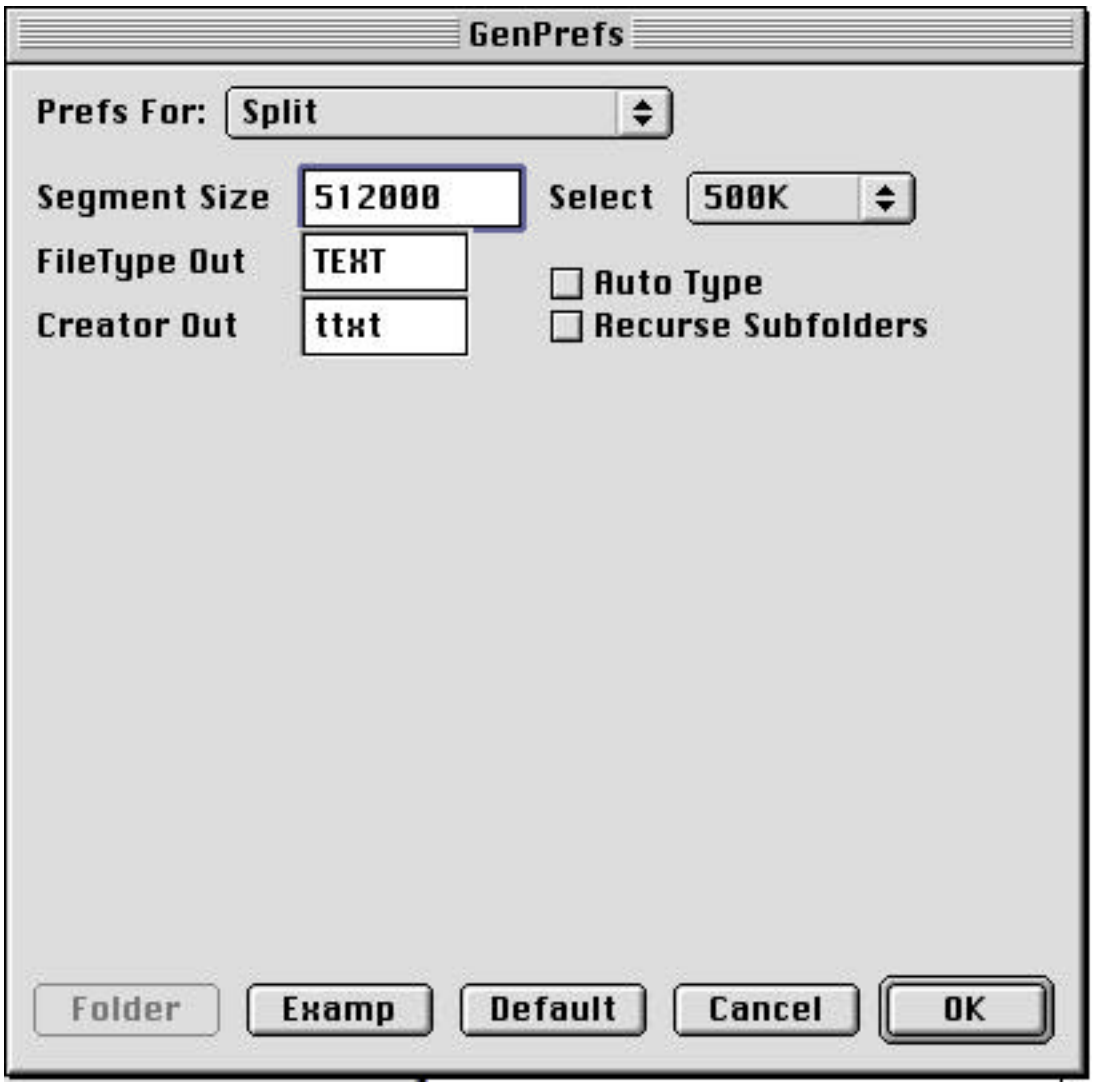
Split

The Split command does the opposite operation as the Join command. It takes one large binary file and splits it into segments of the specified size. A folder is created in the same folder as the original file and which is given a name based on the original filename with the extension removed and the string " *f* " added to its end. For example, if we were to split our previous test movie named "PopeyeSavesOlive.mpg", the folder thus created would be named "PopeyeSavesOlive *f* ". The original file would then be split into segments of the designated size with segment identifiers appended to each segments name. These segment identifiers are of the form ".nnn" (where n is a digit from 0-9) and increment from a starting value of .001 in the same manner as illustrated in the first paragraph of the "Join Folder" command description. These segments are saved to the folder which MNT created for the purpose as described above.

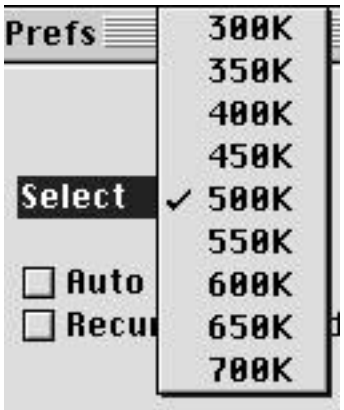
Ops Select dialog for Split Command



Full Prefs dialog for Split Command



Select Popup shown in pulled down state



Segment Size - specifies the size in bytes of the segments to be generated.

FileType Out - specifies the Mac File Type code to be assigned to the generated segments.

Creator Out - specifies the Mac File Creator code to be assigned to the generated segments.

Example Button - brings up a Standard File / Navigation dialog allowing the user to browse for and select an example file of the type and creator desired to be used for the above described file Type and Creator. if the user selects Open in that dialog, that file's Type and Creator are determined and loaded into the File Type and Creator fields described above.

Auto Type - causes MNT to autotype the file segments generated by the Split command rather than to use the type and creator codes entered in the "FileType Out" and "Creator Out" boxes. See section on Automatic File Typing for a full description of this feature. In this case it is primarily just for show since the files themselves cannot be opened, but does avoid the tacky looking situation we have all seen before where files have an empty type and creator and are displayed with only a generic document icon. It also helps to show the user which application owns the file that results when the file segments are combined.

Recurse Subfolders - Normally Split is used with single files. However, if a folder full of files is dropped onto MNT and the Split command selected, all files within that folder will be split into separate subfolders of file segments. Taking this one step farther, if the Recurse Subfolders box is checked and a folder containing files and subfolders of files is dropped onto MNT for splitting, each file in that folder and all of its subfolders (and their subfolders etc) will be split into folders full of file segments in the appropriate locations and manner as described above. Nobody will ever need this unless they are the King of some newsgroup and frequently do massive amounts of binary postings.

Select Popup - (see above) unless they are autistic, most people do not remember the exact odd number of bytes that is equivalent to lets say 550K for example. The de facto standard for mpeg segments is 500K or 512000 bytes each, but some posters like to use other segment sizes. The Select Popup provides a simple means of loading the number of bytes into the Segment Size box for selected even numbered Kbyte values. Simply select that value from the popup menu and its equivalent in bytes will be loaded into the Segment Size box. Segmentation as used for software postings typically uses much larger segments than multimedia postings (usually around 5-10 MB each). Perhaps on some future rainy day I might also add some of these values to the Select Popup menu.

Defaults Button - resets all parameters in this panel to their original default values.

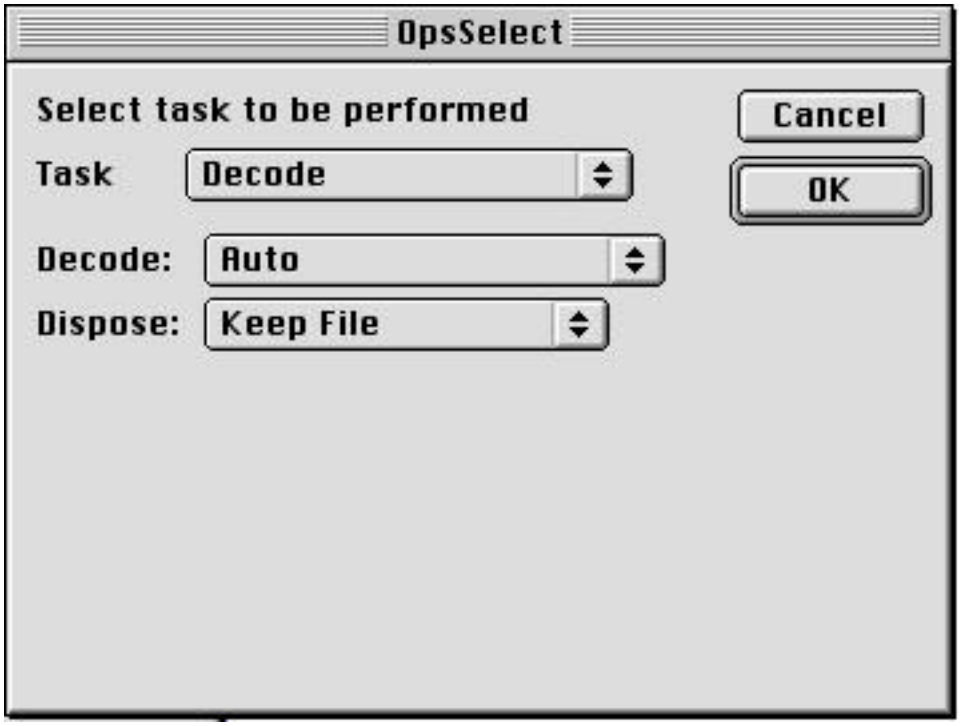
Decode

The Decode command is used to decode binary attachments which have been encoded for transmission using a variety of formats including UUencoding, MIME, AppleSingle, AppleDouble, BinHex, MacBinary and yEnc. Automatic format detection is available covering all of the above. It is even possible to decode (and encode) files in formats that nobody would ever consider, for example a MIME AppleDouble encoding with the data fork UUencoded and the AppleFile yEnc encoded.

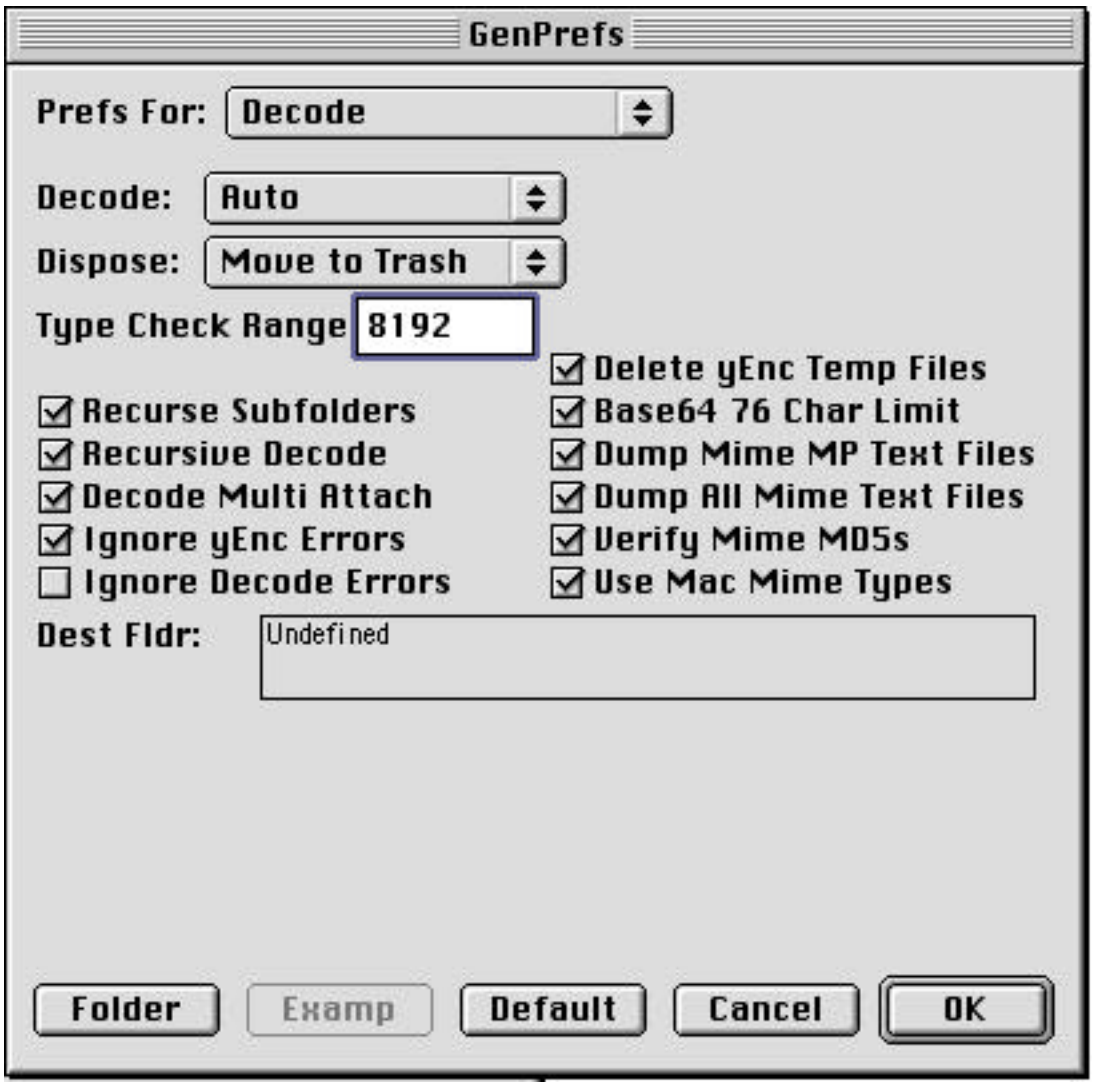
For MIME encodings, the decoder recognizes the following Content Transfer Encoding types 7 Bit Text, 8 Bit Text, Binary, Quoted Printable, Base64, UUEncode, X-UUEncode, X-UUE (all three are the same but have different tags) and X-yEnc (although this format does not yet officially exist). Automatic CTE detection covers all of the above. Currently, no distinction is made between 7 and 8 Bit Text encodings and Binary encodings - all are treated the same with data copied unchanged directly to the output file. See “Miscellaneous Notes” chapter for additional info on the new MIME functions.

It is my belief that all Mac users should consider Stuffit Expander to be **THE** reference decoder of choice. That is to say that when other decoders produce questionable results, the user should always rely on the results obtained from Stuffit Expander over all others. However, Stuffit Expander is not always the most convenient or fastest decoder to use compared to others including MNT so the user may prefer to use decoders such as MNT until problems arise and only then to fall back upon the results obtained from Stuffit Expander. In practise, I have found files which MNT will not decode and which Stuffit Expander will decode, but I have also found just the opposite in cases of files which MNT will decode but which Stuffit Expander will not. Each piece of software has its own inherent quirks and complex data can bring those to light beyond any developer’s testing abilities. The wise user will make the most of all tools which are at his diposal.

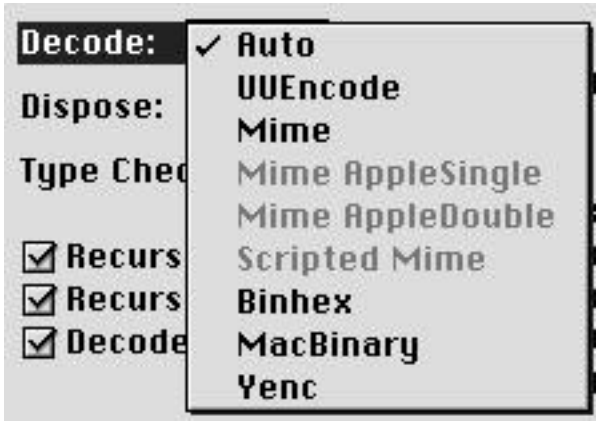
Ops Select dialog for Decode Command



Full Prefs dialog for Decode Command (described in full detail below)



Decode Popup shown in pulled down state



Dispose Popup shown in pulled down state

Decode Popup - this popup selects the format to be used to decode the dropped file(s)/folder(s). Normally you will want to put it on "Auto" and just leave it there always (the default condition). In the unlikely event that you should have a file whose contents are ambiguous and lead the format detector to think it is one type of format when in reality it is something else, you can temporarily set the Decode Popup to the proper format to allow you to decode that file. However, I have tried to cover all the bases to keep the format detector from being fooled so you are unlikely to need to change this setting very often if ever. The main case where this would ever occur is if some message has an undesired MIME header with no MIME content followed by a file encoded by some other method. In such cases you would simply set the Popup to the correct format, decode that file, then reset it to Auto once again.

Dispose Popup - this popup selects the desired method of disposition of input files which have been correctly decoded with no errors.

Keep File	File is not moved
Delete File	Delete File in place
Move To Trash	Move File To Trash
Move To Folder	Move File To Folder selected using Folder button and shown in Disp Fldr box

Type Check Range - before decoding each file, an automatic format detector scans each file to determine the encoding format used with that file. Since all of the information needed to determine a file's encoding method is located at the beginning of the file, it is wasteful and slow to scan the entire file, especially with large files. If a non zero positive value is entered into the Type Check Range box, then this scanning is limited to that many bytes following the beginning of the file (or beginning of each part in the case of multipart files). Setting the value to zero causes the full file to be scanned. Typically 4 to 8 kbytes is enough to cover all of the info that is needed to determine a file's format.

Recurse Subfolders - if checked, causes files within subfolders of any dropped folders to be processed in addition to the files within the dropped folder itself. Otherwise only the files in the dropped folders themselves are processed.

Recursive Decode - sometimes people will double encode binaries, either by accident or intentionally. The recursive decode option causes MNT to attempt to decode each file again after being decoded from the input file. There is no limit on the number of times that a file may be reencoded without MNT being able to decode it other than the amount of memory available for use in this process. This can get to be a very memory hungry operation with each level of recursion but you are unlikely to encounter anything deeper than a double encoding and seldom that. It is best to keep leave option turned on. See Buffer Size discussion under "The Prefs Dialog" subsection for info on memory requirements for recursive decode. For those who are unfamiliar with the word, a "recursive" operation is one which reflects or folds back on itself. In programming parlance it typically refers to a function which calls itself, sometimes repetitively.

Decode Multi Attach - if checked, then after decoding one attachment within an input file, MNT will continue to search for and attempt to decode additional attachments within that same file. If not checked, then only the first attachment will be decoded within each file.

Strict yEnc Filenames - no longer needed, now removed

Use yEnc Temp File - no longer needed, now removed

Ignore yEnc Errors - many posters in the PC world continue to use software which encodes the yEnc format incorrectly. Some programs do not bother to deal with yEnc errors at all. Others generate posts with invalid CRCs (common on mpg .000 files) and multiple part instances. Combine it all and you will get corrupt data without even knowing it. However in those cases where partial data loss is not a serious problem (such as with mpeg movies and other non-executables etc), it is possible to get a decoded binary out of it (although possibly corrupt) by checking this option. In other words, this option causes MNT to behave like other Mac programs as far as yEnc errors goes. if you start to get a lot of yEnc errors and failed decodes, try turning this option on. Otherwise leave it off.

Ignore Decode Errors - similar to Ignore yEnc Errors but causes ALL decode errors to be ignored. Unlikely to ever be needed but its here just in case.

Delete yEnc Temp Files - causes any remaining yEnc Temp files in a dropped folder or parent folder of a dropped file to be deleted following the decode of each dropped item (file or folder as the case may be). Should not be used if you expect to be decoding yEnc multipart files with parts arriving at separate, unknown times. See “Comments Regarding yEnc” under “Miscellaneous Notes” chapter for a description of the need for this option.

Base64 76 Char Limit - Base64 encoding limits encoded lines to a length of 76 characters (RFC 2045). Selecting this option results in an error condition if this limit is exceeded. This option should normally be left checked.

Dump Mime MP Text Files - this is your spam filter. Frequently, multipart MIME encodings have a descriptive text part included. These are being increasingly used for advertisements for every Tom, Dick and Harry’s “pay me” web site and get rich quick schemes. Checking this option will cause all multipart text files to be immediately deleted on decode.

Dump All Mime Text Files - similar to option above, but is effective for ALL text encodings and not just those in multipart messages.

Verify Mime MD5s - causes MNT to compute the MD5 checksum for all decoded binaries and to compare it to the value received with the message (if any). If the received value is non-zero and the two values do not match, then the decoded file is deleted and an error is reported to the log file if enabled.

Use Mac Mime Types - some MIME encodings include the x-mac-type and x-mac-creator tags which contain Finder filetype information for the file being transmitted. Selecting this option causes MNT to use this information (when present) to type the decoded file. Otherwise it is always typed by MNT’s automatic filetyping function based on the filename’s extension.

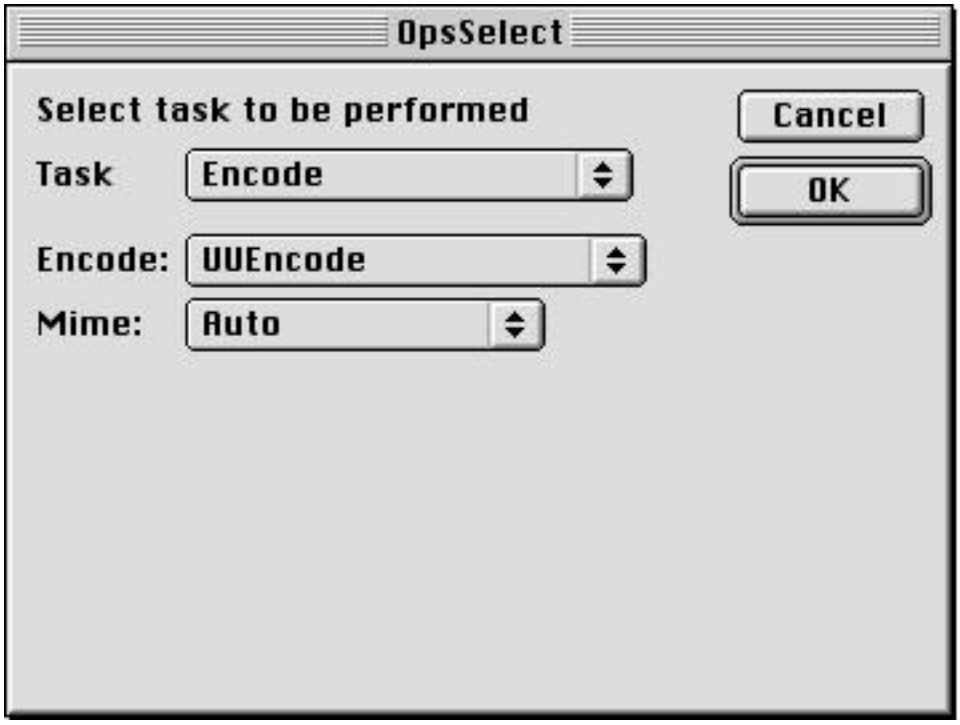
Dest Fldr - displays full path to folder specified as move to folder for file disposition (see info on Dispose Popup). If no folder has been specified, this box displays “Undefined”.

Folder Button - brings up Standard File or Navigation dialog as appropriate for the user to navigate to and select the folder to be used for file disposition.

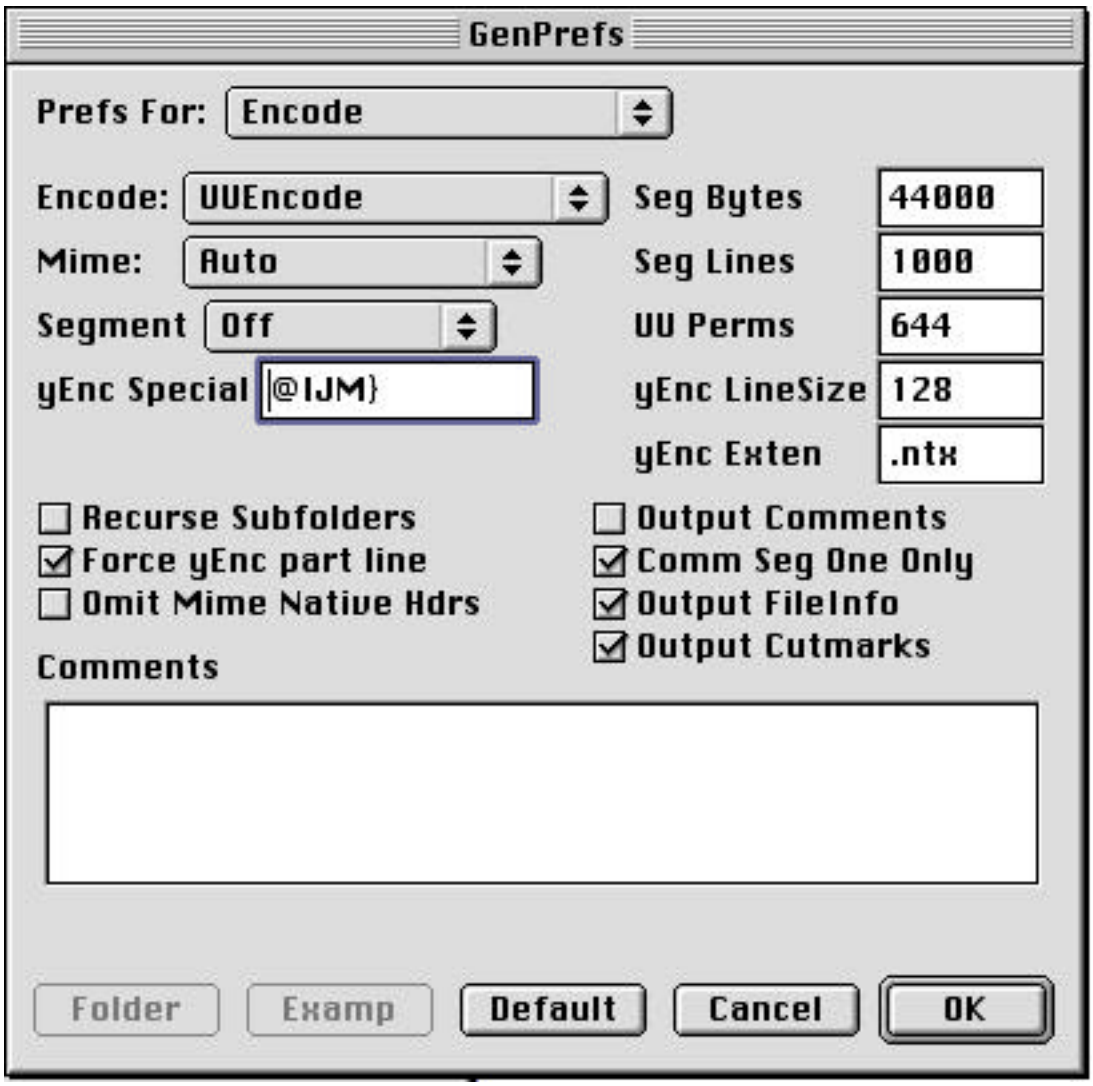
Encode

The Encode command is used to encode binary attachments into a variety of formats including UUencoding, MIME, AppleSingle, AppleDouble, BinHex, MacBinary and yEnc. For MIME encodings, a variety of Content Transfer Encoding types are available including Auto, 7 Bit Text, 8 Bit Text, Binary, Quoted Printable, Base64, UUEncode, X-UUEncode, X-UUE and X-yEnc. It is even possible to encode files in formats that nobody would ever consider, for example a MIME AppleDouble encoding with the data fork UUencoded and the AppleFile yEnc encoded. However, not all of these CTE types are implemented as separate types. In particular, 7 and 8 bit text are treated identically as are UUEncode, X-UUEncode and X-UUE. The file(s) to be encoded or folders containing these files are simply dropped onto MNT. The encoded files are then output to the same folder as the original with an appropriate extension appended to the filename depending on the encoding method used (namely .uu, .mime, .hqx, .bin, .ync or as user specified for yEnc only).

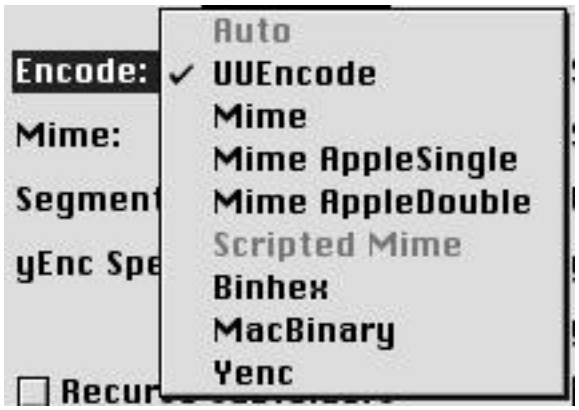
Ops Select dialog for Encode Command



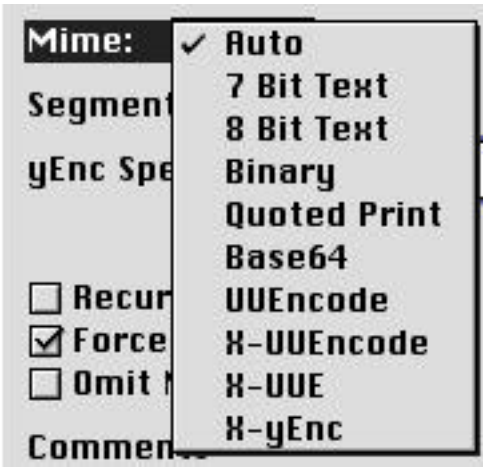
Full Prefs dialog for Encode Command (described in full detail below)



Encode Popup shown in pulled down state



Mime Popup shown in pulled down state



Segment Popup shown in pulled down state

Encode Popup - specifies the basic format to be used for encoding.

Mime Popup - specifies the Mime Content Transfer Encoding (CTE) type to be used for Mime encodings. Applicable to Mime only.

Segment Popup - specifies the segmentation option to be applied to the encoded file.

Off	the output file will not be segmented
By Bytes	the output file will be generated in segments of nnn bytes each where nnn is the value specified in the Segment Bytes box.
By Lines	the output file will be generated in segments of nnn lines each where nnn is the value specified in the Segment Lines box.

yEnc Special - allows the user to specify which characters will be treated as yEnc special (or escaped) characters. These special characters are always entered as the ASCII character represented by the character's hex code plus $40 (64 decimal) as these special characters are encoded. The standard special characters defined by yEnc are as follows, but occasionally a file will be encountered which contains special characters other than the 5 standard ones (including the standard yEnc test files). The special characters used are transparent to the decoder but the encoder can choose to escape any characters which may present problems with the particular transmission method in use. Unless absolutely needed, this field should be left at the default setting and under no circumstances should you ever remove any of the standard special characters from the list. If you do so, then yEnc can be guaranteed not to work.

Char	Hex	Hex Rotated	ASCII Rotated
NULL	00	40	@
Tab	09	49	I
LF	0A	4A	J
CR	0D	4D	M
=	3D	7D	}

Recurse Subfolders - if checked, causes files within subfolders of any dropped folders to be processed in addition to the files within the dropped folder itself. Otherwise only the files in the dropped folders themselves are processed.

Force yEnc Part Line - for single part encodings, yEnc normally does not include the yPart line. However many posters use software which always does provide this line. Selecting this option causes MNT to act the same with the yPart line always encoded.

Omit Mime Native Headers - the Mime format contains enough information to define a file exclusive of all the various appurtenances added by other formats to accomplish the same purpose (for example the begin line in uuencoding and the yBegin, yPart and yEnd lines of yEnc). When these formats are used as Mime CTE types, this information is superfluous and often omitted. Checking this option causes the MNT Mime encoder to omit such additional info lines. The MNT decoders will accept it either way.

Segment Bytes - specifies the number of bytes per segment to be used with segmentation by bytes.

Segment Lines - specifies the number of lines per segment to be used with segmentation by lines.

UU Permissions - specifies the octal code to be output by the encoder in the permissions field of the UU header (e.g., begin 644 filename.jpg). It is most commonly a value of 644 although some Unix systems will output other values. It is only used by Unix systems anyhow.

yEnc Line Size - allows the user to enter the base line size, x where all lines other than the last line are either x or x + 1 characters in length. The most commonly used value is 128 (with 129 for x + 1). The yEnc format permits maximum line sizes ranging from 63 to 998 characters. In practical use, however, Pascal strings are limited to 255 characters so for compatibility with programmers who happen to use Pascal, a more practical limit is 254 characters. Others have suggested lines of 63/64 characters, so until everybody makes up their minds, you can choose whatever values that you wish to use. A value of 128 characters is suggested for now.

yEnc Extension - likewise, nobody has really decided what the proper extension should be used for yEnc encoded files either. File extensions are DOS nonsense anyhow but MNT does make use of them since they appear to be here to stay. Its odd how the Wintel crowd insists on living in the past, isn't it? Anyhow, the most commonly used extension for yEnc seems to be .ntx although I have also seen .ync used too (my personal favorite is .yy which corresponds with .uu for uuencoding). Use whatever you want to use by entering it in the yEnc Extension box. If no extension is specified, then .ntx will be used.

Output Comments - if checked, causes user specified comments in Comments box to be output prior to the UU header in uuencoded files.

Comments Seg One Only - if checked causes above comments to be output to segment one only of multisegment UU encodings.

Output FileInfo - if checked causes descriptive file info (size etc) to be output prior to uuencodings.

Output Cutmarks - if checked causes UU editing cutmarks to be output with uuencodings.

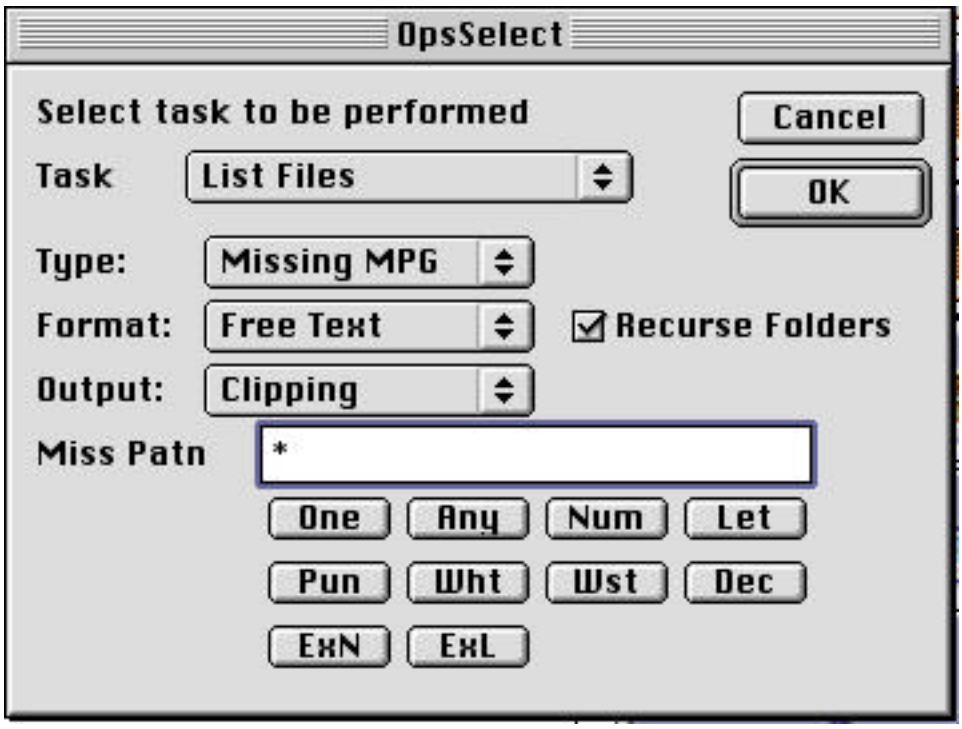
Comments - text entered here is used as user specified comments to be output with with UU encoded files subject to the conditions described above.

List Files

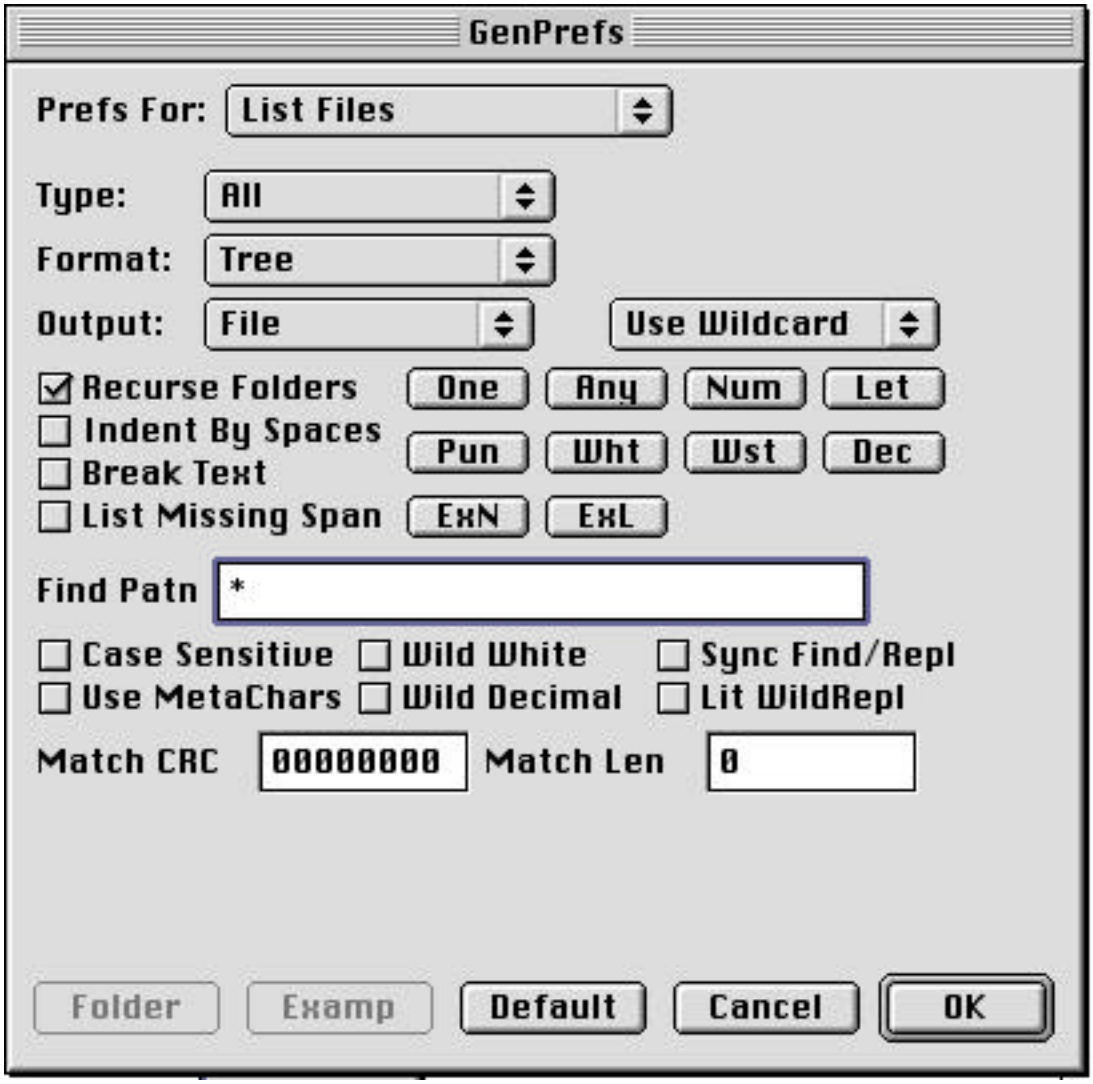
The List Files command is a multifunctional command which lists those files or folders of files dropped onto MNT based on a variety of criteria and output in a variety of formats in several forms. (Now if that isn't a catch-all statement, I don't know what is). The truth is that several of MNT's commands like List Files, File Tools and Folder Tools offer a very wide range of services depending on the options which the user selects making it difficult to give an overall description of the command as a whole. Perhaps the best overall description that could be given would be to say that List Files can make a variety of lists of some or all of the files dropped onto MNT.

The user should also see the separate document on "Pattern Matching" for additional info that applies to some of these commands.

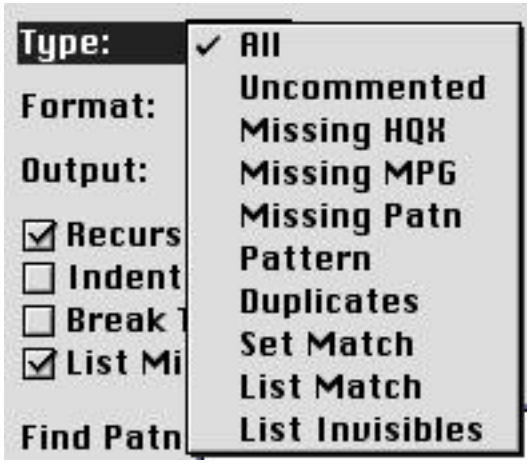
Ops Select dialog for List Files Command



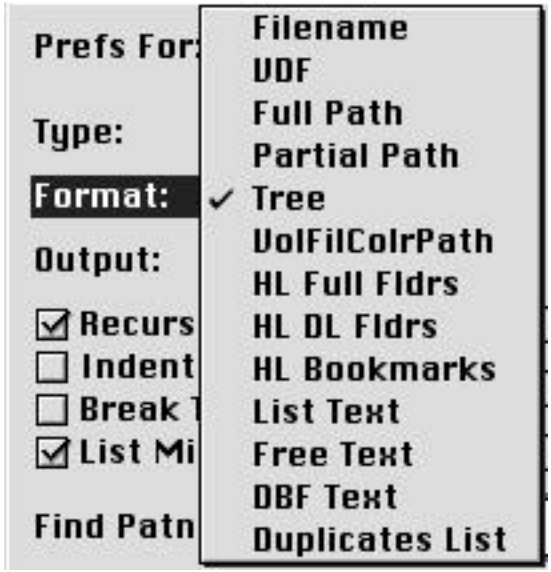
Full Prefs dialog for List Files Command (described in full detail below)



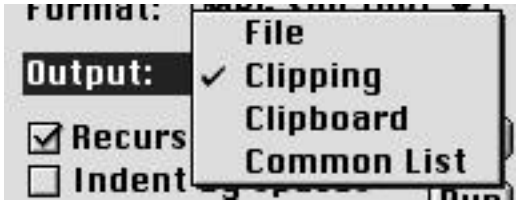
Type Popup shown in pulled down state



Format Popup shown in pulled down state



Output Popup shown in pulled down state



Matcher Popup shown in pulled down state

Type Popup - specifies the overall type of File List operation to be performed

All	All files dropped onto MNT will be listed
Uncommented	Only those files containing no Get Info comments or those files containing only an internet IP number will be listed.
Missing HQX	Only those files that are missing from folders containing a series of Stuffit file segments named by the usual format will be listed
Missing MPG	Only those files that are missing from folders containing a series of MPEG movie segments named by the usual format will be listed (Identical to Missing HQX except for output segment numbering... MPG segs have leading zeros).
Missing Patn	Only those files that are missing from folders containing a series of numbered files will be listed. The numbers by which the missing parts are determined must be specified by a wildcard pattern including the numeric extraction token (ExN) in those positions from which this numeric information is to be extracted from each filename. Requires Wildcard Pattern Matcher selection. See document on Pattern Matching.
Pattern	Only those files whose names match the provided pattern string will be listed. May be used with any pattern matcher. See document on Pattern Matching.

Duplicates — Only those files which are identical in Data Fork size and content to at least one other file in the set dropped onto MNT will be listed. The files' Resource Forks are ignored. Time consuming and computationally intensive but extremely useful in rooting out those duplicate files that tend to accumulate, especially in folders full of pictures and similar downloaded files etc. See Duplicates List format below for further info.

Set Match — Drop a single file onto MNT and select Set Match to use that file as the reference file for the List Match command below.

List Match — Similar to Duplicates list with a different twist to it. List Duplicates compares all files dropped against all other files and lists Duplicates found. List Match uses the one file previously tagged by the Set Match command above as a reference file. The content of all files/folders of files dropped onto MNT is then compared against this reference file. All such matches are then listed. Only the data fork is examined. The lengths are compared before the CRC with CRCs computed only for those files of matching length so this command is much faster than a Duplicates List on a given folder of files. Lets say that you have a jpeg image file and that you know there are duplicates somewhere in a group of folders, perhaps also having different filenames. You would use List Match to quickly find them all.

List Invisibles — Lists only those files which are invisible in the Finder.

Format Popup - designates the format to be used for the output file list

Filename — Only the filename of the files will be listed

VDF — Output the vRefnum, dirID and filename of the listed files. All files stored using the MacOS HFS or HFS+ file systems can be uniquely designated by three parameters. These three parameters are normally of interest to programmers only, but I have found that in some cases it helps some users greatly to understand just a little bit about the nuts and bolts behind that MacOS file system. The VDF format outputs these three parameters for each file listed.

- vRefnum — Volume Reference Number, a small negative number that designates the mounted volume containing the file. vRefnums begin at -1 for the startup volume and decrease by one for each volume that is subsequently mounted.
- dirID — Directory ID (or Parent ID), a positive number (typically large) that is assigned to each directory or folder when it is created. It represents the ID of the folder containing any given file or folder (or its parent folder).
- filename — the name of each file or folder.

Full Path — Outputs the full pathname for each file listed.

Partial Path — Outputs the partial pathname relative to the folder dropped for each file listed.

Tree — Outputs an indented tree like structure for the files listed showing the hierarchy of those files and the folders containing them. Each folder listed is also tagged with its Directory ID in parentheses. (This is the Directory ID of the folder itself and not of its parent folder).

VolFilColrPath — Outputs a tab-delimited list of the following parameters for each file:

- Volume Name
- Filename
- Color (the Finder label assigned to it represented by a single digit number ranging from 0-7).
- Full Pathname

HL Full Fldrs — A highly specialized list format used with Hotline downloads to output a list of files within a folder of subfolders arranged by category. This list is a tab-delimited list of the following parameters suitable for direct import into a database for file cataloging. These parameters are:

- Full pathname of file's parent folder
- Filename
- File's logical size in bytes

HL DL Fldrs — Another highly specialized list format used with Hotline downloads to output a list of files within a single folder. This list is a tab-delimited list of the following parameters suitable for direct import into a database for file cataloging. These parameters are:

- Full pathname of file's parent folder
- Filename
- File's logical size in bytes
- File's modification date
- File's modification time
- File's Get Info Comments

HL Bookmarks — Takes a folder of Hotline Client bookmarks and lists the following parameters for each of them in a tab-delimited list suitable for database entry.

- Server name
- Server IP Number
- Account login username (if present)
- Account login password (if present)

List Text — Used with any of the 3 Missing files items on Type Popup. Outputs information as a list with each individual entries on successive lines (subject to List Missing Span option below). Requires "Missing ***" Type popup selection.

Free Text — Used with any of the 3 Missing files items on Type Popup. Outputs information as continuous free form text with all individual entries on the same line (subject to Break Text and List Missing Span options below). Requires "Missing ***" Type popup selection.

DBF Text — Used with any of the 3 Missing files items on Type Popup. Outputs information as a series of tab delimited text records (one for each common set of items) suitable for import into any database manager. This is a highly specialized list included by user request and typically used for MPG segments but also usable with any numbered group of files. Requires "Missing ***" Type popup selection. The order of output fields for each record is as follows:

- Movie Name
- Segment Size in Kbytes
- Logging Date (i.e. Date when list is run... current date right now)
- Logging Time
- Missing Segments (open text record of indeterminate size - do not use an alpha field of known size)
- Highest Segment Number Encountered

Example of a typical record follows:

MickeyMowsTheLawn 597K 8/21/02 17:42:34 099-147 812

Duplicates List — Scans the files and folders of files dropped onto MNT and lists to the output those files which have identical content to any of the other dropped items. Requires "Duplicates" Type popup selection. This is a very powerful tool for file comparison. It is also computationally intensive and rather slow compared to other MNT commands. NOTE that duplicate filenames are not considered in this command in any way. Likewise, the Resource Fork is not examined or taken into consideration (although this may very well become a future option). This command looks only at the actual contents and size of each file's data fork.

In order to perform this operation within an acceptable amount of time, MNT uses a simple trick to identify a file's contents. A table is built containing each file's Data Fork CRC and size values. Each entry in this table is then compared to all other entries and those files with identical values are listed. This method, while fast, is not exact. It will positively identify **ALL** true duplicates, however, there is a one in 4 billion chance that any given file's CRC value will match that of another totally different file. The file size comparison was added to further reduce this already tiny chance to the point of insignificance.

The time required for a List Duplicates operation varies depending on a number of factors including processor speed, total number of files, combined file size and to a much lesser extent, the number of Duplicates found. For example, with a 800 Mhz G4 it takes about 10 seconds to run a Duplicates list on a folder of 930 files totaling 60 MB. A list on a folder of 3145 files totaling 204 MB takes 41 seconds. Generally speaking the time required will increase geometrically with the number of files and linearly with the average file size.

List Duplicates works similarly to the Move Duplicates command under File Tools with one exception. List Duplicates will list **ALL** duplicates found including the first instance of each duplicated file. Move Duplicates on the other hand will always keep the first instance of each duplicated file and will move all of the rest.

With version 1.2.1, the duplicates handling code for both List Duplicates and Move Duplicates has been rewritten in a simpler more efficient form. The by product of this simplification is that totally annoying, but previously necessary "Just Spinning Wheels" phase has now been eliminated. I'm sure that all of you purists out there will appreciate this.

There might be some cases where it would be desirable to examine and compare part or all of two files actual data bytes. For this reason I will probably add a slow Duplicates List command in future versions which uses a byte by byte comparison rather than the CRC method. However this is not on my immediate list of things to do.

Output Popup

File	Outputs info to a text file in the same folder as the files/folders dropped. This file is named "xxx Filelist" where xxx is the name of each files/folders dropped.
Clipping	Outputs info to a text clipping in the same folder as the files/folder dropped. This clipping is named "xxx Filelist" where xxx is the name of each files/folders dropped.
Clipboard	Outputs info to the clipboard for subsequent paste into any other application.
Common List	Outputs info from all items dropped to a common text file in the same folder as the files/folders dropped. This file is named "MNT Filelist". For a single dropped item, Common List works identically to the File output option except for the name of the output file.

Matcher Popup — Selects matcher to be used for text matching

Use Wildcard	Text matching will be done using Wildcard matcher.
Use Regexp	Text matching will be done using Regexp regular expression matcher.
Use Regexs	Text matching will be done using Regexs regular expression matcher.

Recurse Subfolders - if checked, causes files within subfolders of any dropped folders to be processed in addition to the files within the dropped folder itself. Otherwise only the files in the dropped folders themselves are processed.

Indent By Spaces - Indent the tree list by spaces rather than by tabs. Applies only to the Tree output format. Each level of folder recursion is shown in the list by an indentation level of one tab character. If Indent By Spaces is checked, each indentation will be three space characters rather than one tab character.

Break Text - For the continuous text output formats (MPG Seg Text and MPG Patn Text) checking the Break Text box causes the continuous text to be broken into lines of no more than 72 columns.

List Missing Span - For all MPG missing lists formats, each individual segment is normally listed individually (e.g. 005, 006, 007, 010, 012 etc). Checking the List Missing Span box causes all consecutive missing segments to be listed as a span rather than individually (e.g. 005-007, 010, 012 etc).

Match CRC - Used by Set Match / List Match. Reference File CRC obtained by Set Match operation and used by List Match is displayed here. It may also be entered manually but there is no reason why anyone would ever want to do this.

Match Len - Used by Set Match / List Match. Reference File Length obtained by Set Match operation and used by List Match is displayed here. It may also be entered manually but there is no reason why anyone would ever want to do this.

Buttons - One, Any, Num, Let, Pun, Wht, Wst, Dec, ExN, ExL - all are described in separate document on Pattern Matching

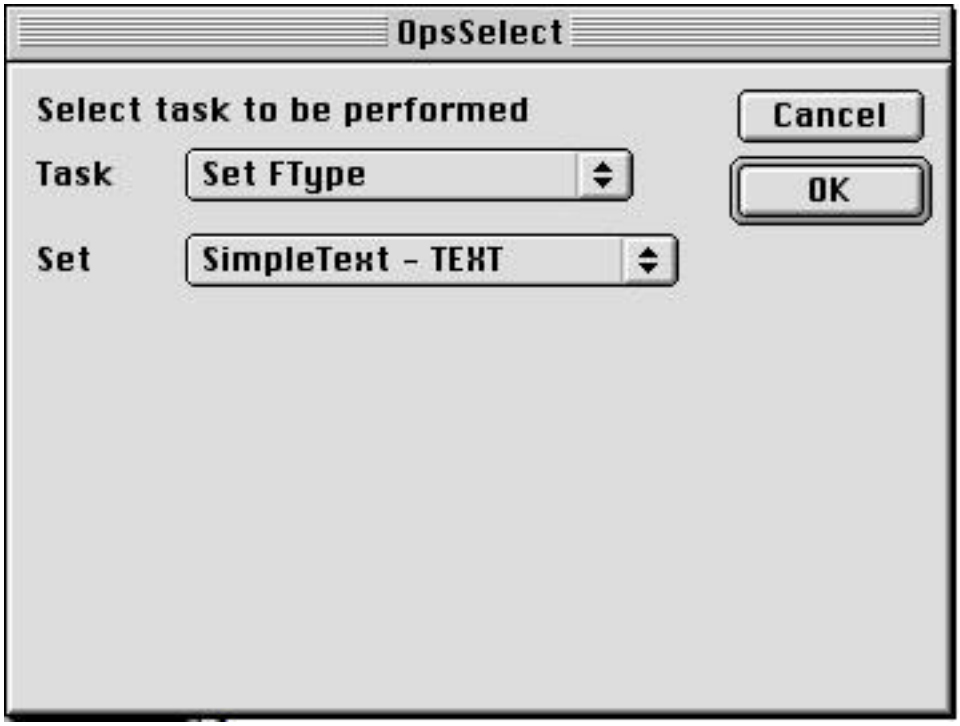
EditText fields - Find Patn - all are described in document on Pattern Matching

Checkboxes - Case Sensitive, Use MetaChars, Wild White, Wild Decimal, Lit Wild Repl - all are described in document on Pattern Matching

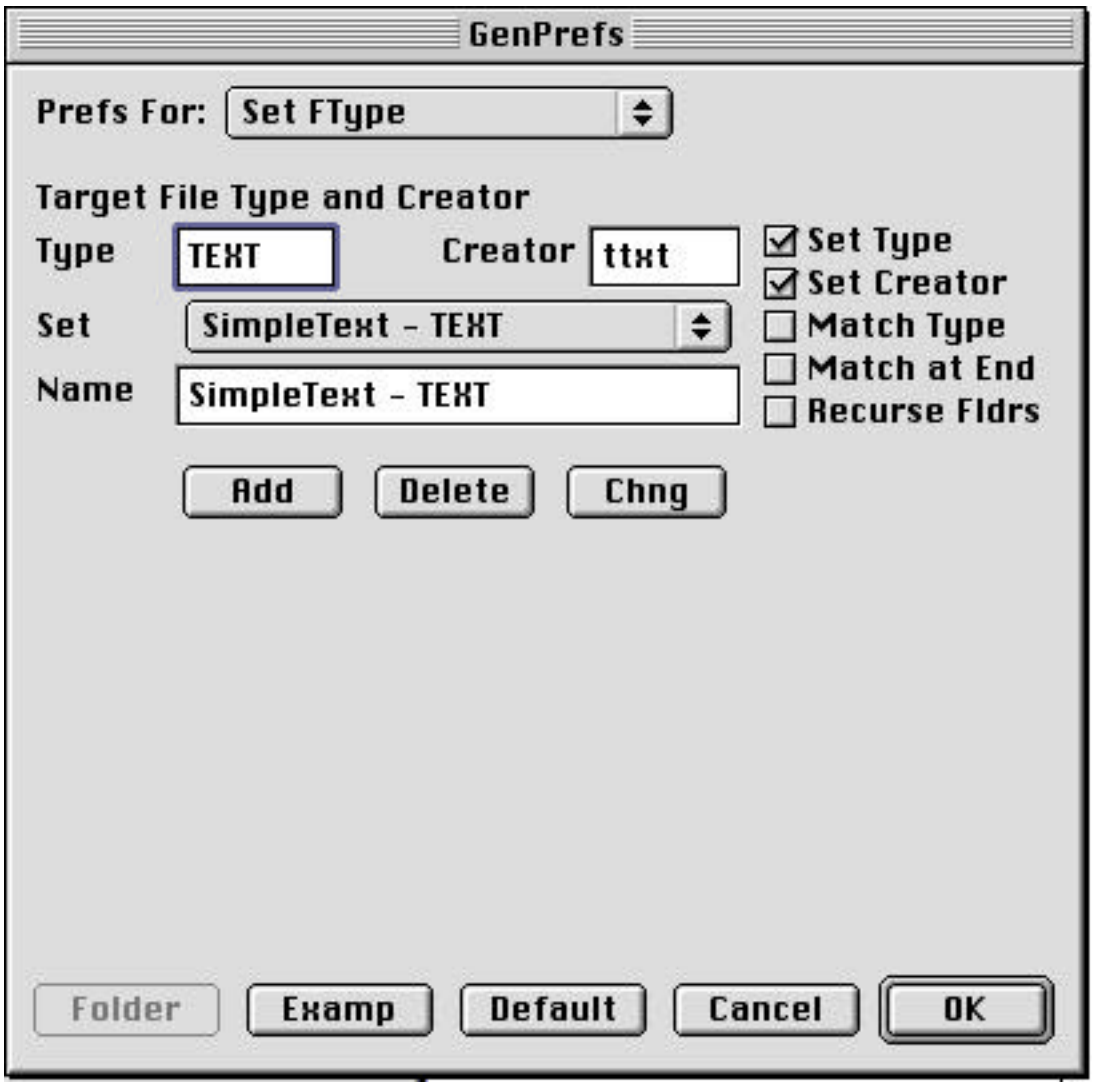
Set FType

The Set FType command sets the Finder filetype and creator of all files, folders of files and subfolders of files (if recursion is enabled) dropped onto MNT to the type of file selected by the Set popup menu. If there is any command in MNT where the newbie can do some real damage through absent minded experimentation, then this is it. Do not use this command unless you know what you are doing and really want to change a file's type and/or creator. For the user who knows what he is doing, however, this command provides a very easy method to quickly set a variety of filetypes without having to have a folder full of all kinds of autotypers. The types available on the Set popup menu are fully editable. See chapters on the Edit menu (Edit FT List item) and File Type Editor for further details.

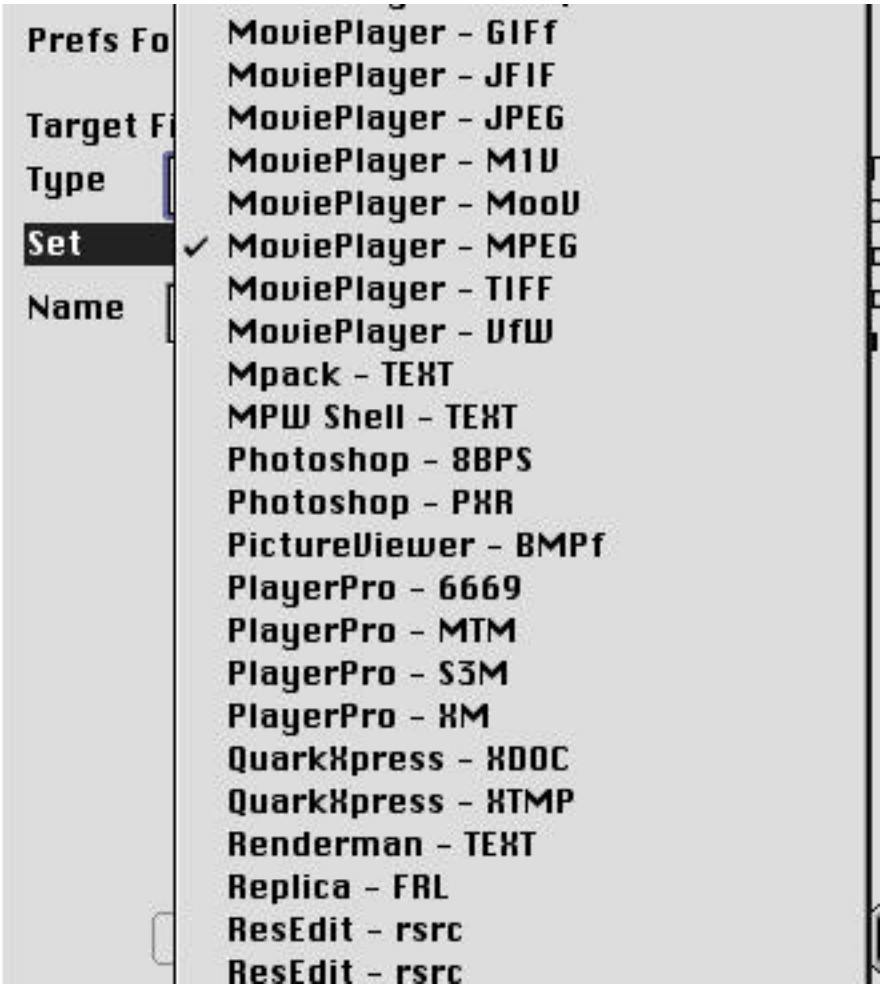
Ops Select dialog for Set FType Command



Full Prefs dialog for Set FType Command (described in full detail below)



Set Popup (partial but typical) shown in pulled down state



Set Type - if checked, Set Ftype will set the Finder filetype of all dropped files and folders of files.

Set Creator - if checked, Set Ftype will set the Finder file creator of all dropped files and folders of files.

Match Type - if checked, Set Ftype will only set info for those files whose Finder filetype matches the type shown in the Type box.

Match at End - N/A for Set Ftype

Recurse Subfolders - if checked, causes files within subfolders of any dropped folders to be processed in addition to the files within the dropped folder itself. Otherwise only the files in the dropped folders themselves are processed.

Type Box - displays the Finder filetype to which dropped files are to be set, contingent on the other options above.

Creator Box - displays the Finder file creator to which dropped files are to be set, contingent on the other options above.

Set Popup - selects any of a wide variety of file types for display/use by the Set Ftype command.

The types available on the Set popup menu are fully editable. See chapters on the Edit menu (Edit FT List item) and File Type Editor for further details.

Name Box - displays the name of the application owning the Finder type/creator displayed in the Type and Creator boxes.

Add Set Button - Buggy - Appends new set to end of Set popup menu that consists of the Type, Creator and Name currently shown in the edittext boxes by those names.

Del Set Button - Buggy - Deletes the set currently shown on the Set popup menu and redisplays the new selection (next item in the list or previous item if at end of list).

Chng Set Button - Buggy - Changes the set currently shown on the Set popup menu to become the Type, Creator and Name currently shown in the edittext boxes by those names.

Example Button - brings up a Standard File / Navigation dialog allowing the user to browse for and select an example file of the type and creator desired to be used for the current file Type and Creator. if the user selects Open in that dialog, that file's Type and Creator and owning application Name are determined and loaded into the File Type, Creator and Name fields described above.

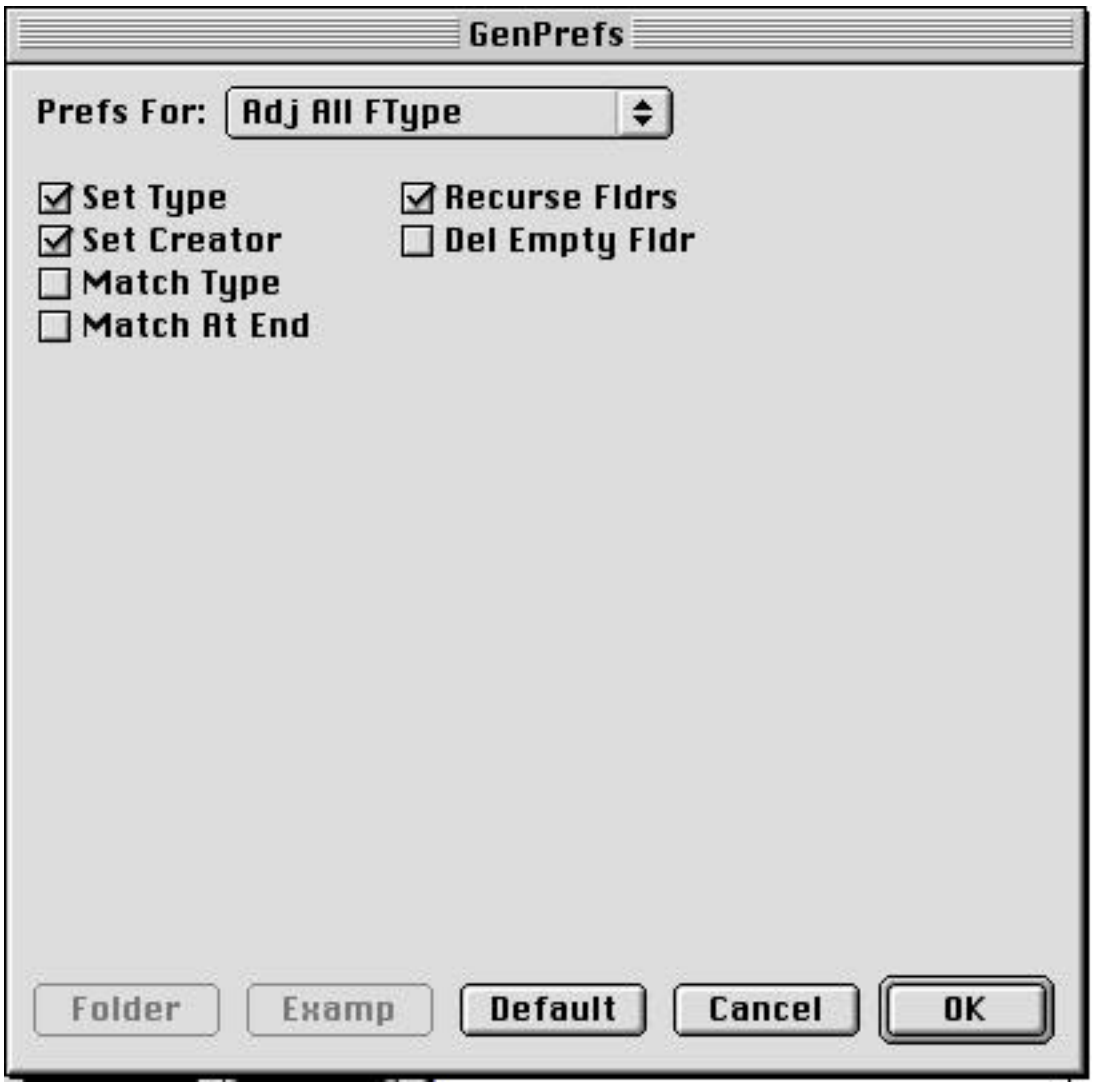
Defaults Button - resets all parameters in this panel to their original default values.

Adj All FType

The Adj All FType command operates somewhat similarly to Set Ftype described above. It sets the Finder filetype and creator of all files, folders of files and subfolders of files (if recursion is enabled) dropped onto MNT to the type of file appropriate to the filename's extension. MNT's Automatic File Typing facility is used to examine each file's name as it is scanned and to determine the appropriate type and creator for that file. See chapters on Automatic File Typing, the Edit menu (Edit Adj List item) and File Type Editor for further details. Lets say that you have downloaded a mixed batch of internet files (pics, movies, text or whatever) and they have a generic filetype (SimpleText filetype for jpegs for instance) or incorrect filetype for your preferred viewer application. Once you have set up the Edit Adj List to your own tastes (a one time operation), you simply drop those files or folders of files onto MNT, select the Adj All FType command and those files will be automatically converted to your preferred filetypes based on the extension in their filenames. This is one of the simplest yet most useful of MNT's many commands.

Ops Select dialog for Adj All FType Command

Full Prefs dialog for Adj All FType Command (described in full detail below)



Set Type - if checked, Adj All FType will set the Finder filetype of all dropped files and folders of files.

Set Creator - if checked, Adj All FType will set the Finder file creator of all dropped files and folders of files.

Match Type - N/A for Adj All FType

Match at End - if checked, Adj All FType will only set info for those files whose extension matches a given extension in the Adj List and only when that extension occurs at the very end of the filename. For example, we may have two files with names such as those shown below:

DonaldDuckQuacks.jpg
DonaldDuckBlushes.jpg.01

If Match at End is not checked, then the extension matcher will find the ".jpg" extension of both files even if it is separated from the filename end in the second case. This is what you want in most cases. However, if Match at End is checked, then the extension matcher will find the ".jpg" extension only in the first case since Match at End requires the extension to be at the very end of the filename.

Recurse Subfolders - if checked, causes files within subfolders of any dropped folders to be processed in addition to the files within the dropped folder itself. Otherwise only the files in the dropped folders themselves are processed.

Del Empty Fldr - if checked, any empty folders found in the process of file scanning will be deleted. This command also exists separately under Folder Tools.

Defaults Button - resets all parameters in this panel to their original default values.

Concatenate

The Concatenate command is similar to the Join command. For those who are unfamiliar with the word, concatenate means to tack an item or series of items onto the end of another item or as defined in the dictionary:

1. To connect or link in a series or chain.
2. Computer Science. To arrange (strings of characters) into a chained list.

This is precisely what the Join command does, however the Concatenate command is somewhat more generalized than the join command which has additional features making it best suited for use with mpeg and/or movie segments. You can turn all those features off in the Join command and it will work a lot like concatenate, but the the Concatenate command has its own set of additional features which makes it separately useful in certain cases. Concatenate takes a series of either files (any kind - it simply treats the data fork as raw data) or clippings (text clippings only) as its input and outputs the combined results to either one common file or one common text clipping (the clipping output is obviously useful for text only).

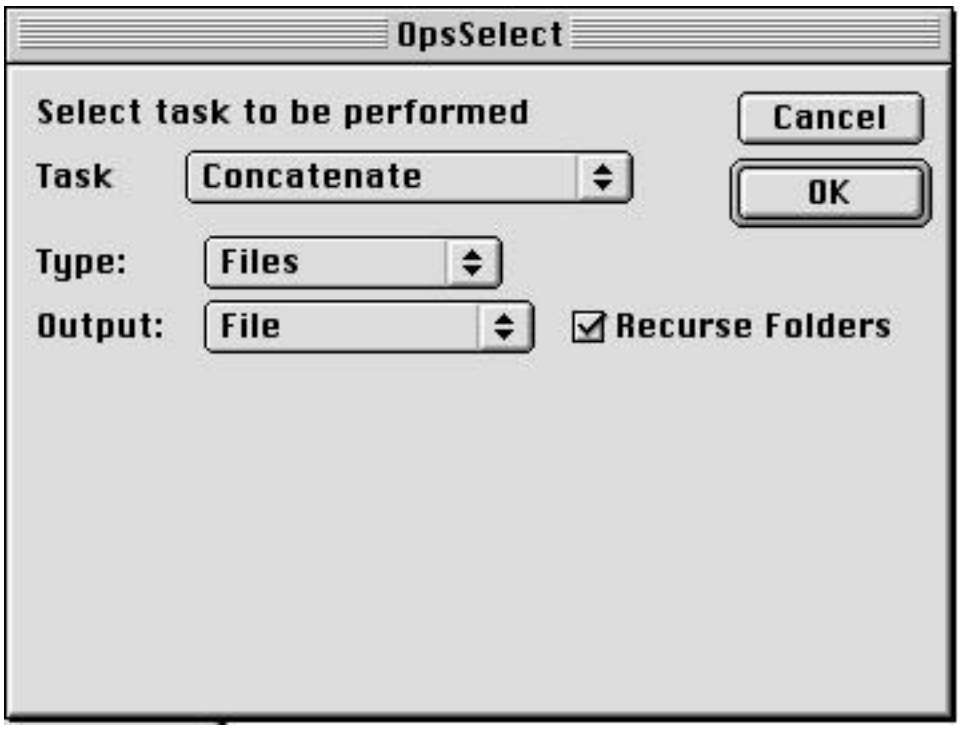
Lets say that you have regular online chats with someone and that you routinely save each exchanged message to a folder somewhere. These itty bitty text clippings will rapidly accumulate and be of little use if you should want to find some remembered point of interest somewhere among them all. It only makes good sense to combine them all from time to time into one single continuous document (this sure makes life easier for the file system too). Concatenate will allow you to do that, although it is not as cleanly simple as it was prior to OS 8.5 when all text clippings were named in numerical order. Text clippings are now named according to the first few characters of text which they contain and if ordered by name (as the file system does and MNT's file scanner follows), they will not appear in the same order as they were created. There are, fortunately, other utilities (one good choice is a handy little program called "A Better Finder Rename" - See Credits for URL) which can take lists of files and rename them according to their creation date. This renamed list of text clippings can then be fed to MNT to be combined via Concatenate into one combined text file. Another use for Concatenate might be if you have a folder full of saved usenet articles or mail on a given topic and wish to combine those files into one single document for archiving.

Concatenate works best with text data (and only for text data in the case of clipping input), but its use with file input is not necessarily limited to text. In the chapter "Miscellaneous Notes" I tell a story about one user who was trying to join two complete mpeg files to create one combined movie. While this is not an appropriate use of either Join or Concatenate, under certain circumstances it just might happen to succeed, in which case either Join or Concatenate could be used to accomplish the task. The user must simply determine that the data to be combined is appropriate for such a combination to begin with. After all, one cannot add 3 apples to 5 oranges and expect to get 8 grapes out of it. As with this kitchen example, the results in such a case are more likely to be one big mess.

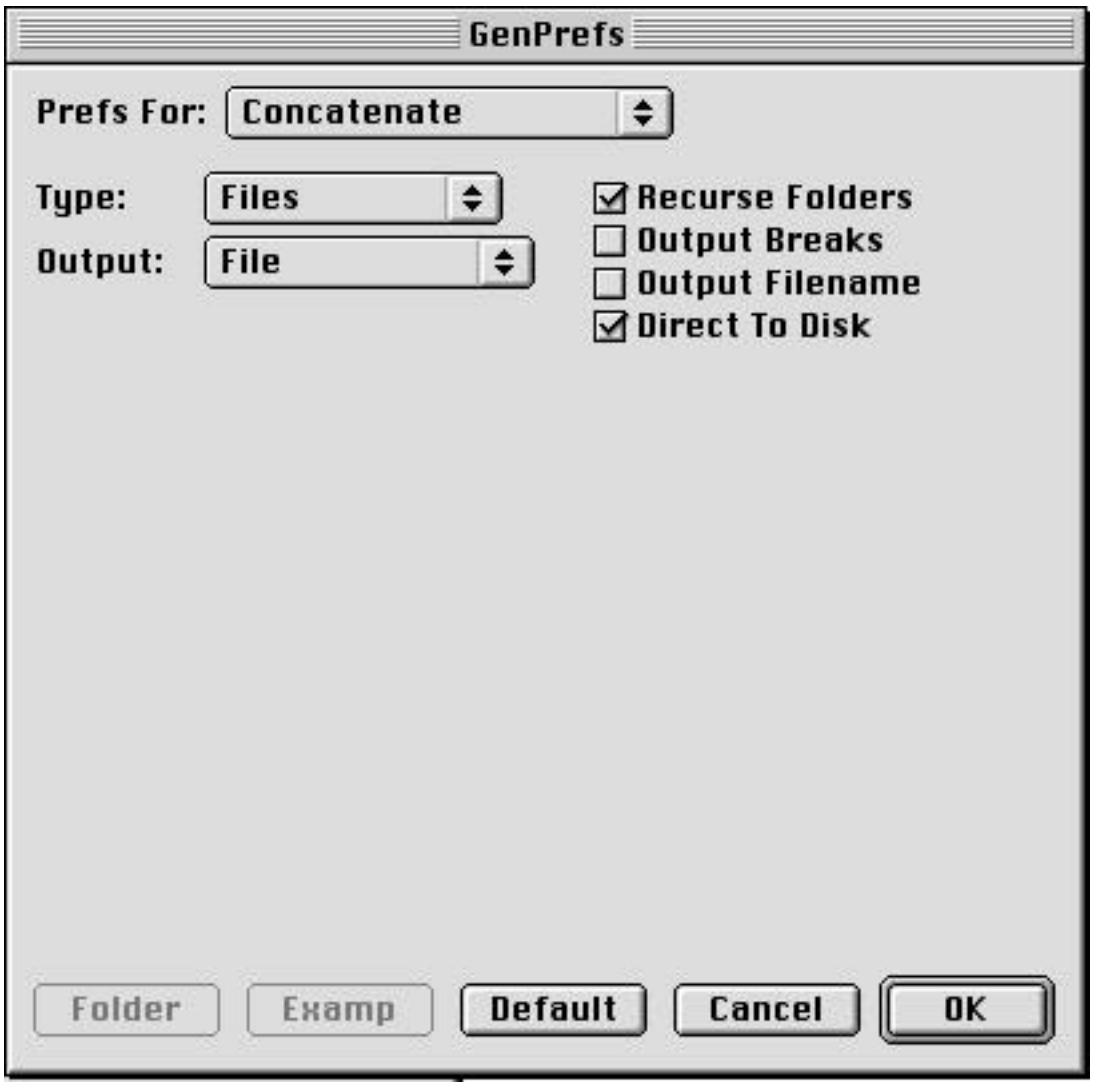
No attempt is made to type the output files generated by Concatenate since the data may be widely varied and no extension is assigned to its output files. The files are typed as Simpletext files for the convenience of its most likely use in joining text files, but the user may easily change that filetype to fit other needs, either by use of Set Ftype or by adding an appropriate extension to the filename and using Adj All Ftype to retype it.

Be sure to read the notes with the description of the Direct To Disk checkbox below in order to understand the two modes in which Concatenate operates.

Ops Select dialog for Concatenate Command



Full Prefs dialog for Concatenate Command (described in full detail below)



Type Popup shown in pulled down state

Output Popup shown in pulled down state

Type Popup - selects the type of input files which MNT will accept. The available choices are:

Files - the data fork of all dropped files is read
Clippings - all dropped files are read as text clippings (if such clipping data exists).

Output Popup - selects the target of the output data. The available choices are:

File - Combined output will be to a file with type and creator as set for the default type and creator on the General prefs panel.
Clipping - Combined output will be to a text clipping.
Clipboard - Combined output will be to the desk scrap (clipboard).

File	Outputs info to a file in the same folder as the files/folders dropped. The type and creator will be set to the default type and creator entered on the General prefs panel. This file is named "xxx Output" where xxx is the name of each files/folders dropped.
Clipping	Outputs info to a text clipping in the same folder as the files/folder dropped. This clipping is named "xxx Output" where xxx is the name of each files/folders dropped.
Clipboard	Outputs info to the clipboard for subsequent paste into any other application.

Recurse Subfolders - if checked, causes files within subfolders of any dropped folders to be processed in addition to the files within the dropped folder itself. Otherwise only the files in the dropped folders themselves are processed. Probably not a good idea with either Join or Concatenate.

Output Breaks - if checked, a horizontal line consisting of hyphens (-----) will be output between each input file or clipping.

Output Filename - if checked, the name of each input file will be output before its contents.

Direct To Disk - if checked, File output will be forced and each input file or clipping will be directly to the output disk file as it is read. Otherwise the combined data is stored in memory until all input is read and then written to the output in one operation. If the total size of all the files to be combined exceeds the memory that you have allocated to MNT, you will get an error -108 (out of memory) before the combine completes. The Direct To Disk option makes it possible to combine files whose total size exceeds the memory allocated to MNT without errors. This option used to be significantly slower than when not checked but this code has now been rewritten and is much faster so it is now the default case.

Defaults Button - resets all parameters in this panel to their original default values.

CRC & Checksums

The CRC & Checksums command computes the user's choice of a variety of CRCs or checksums for each file/folder of files dropped onto MNT and outputs the results in one of several user chosen formats. Most of these functions provided in MNT are CRC functions.

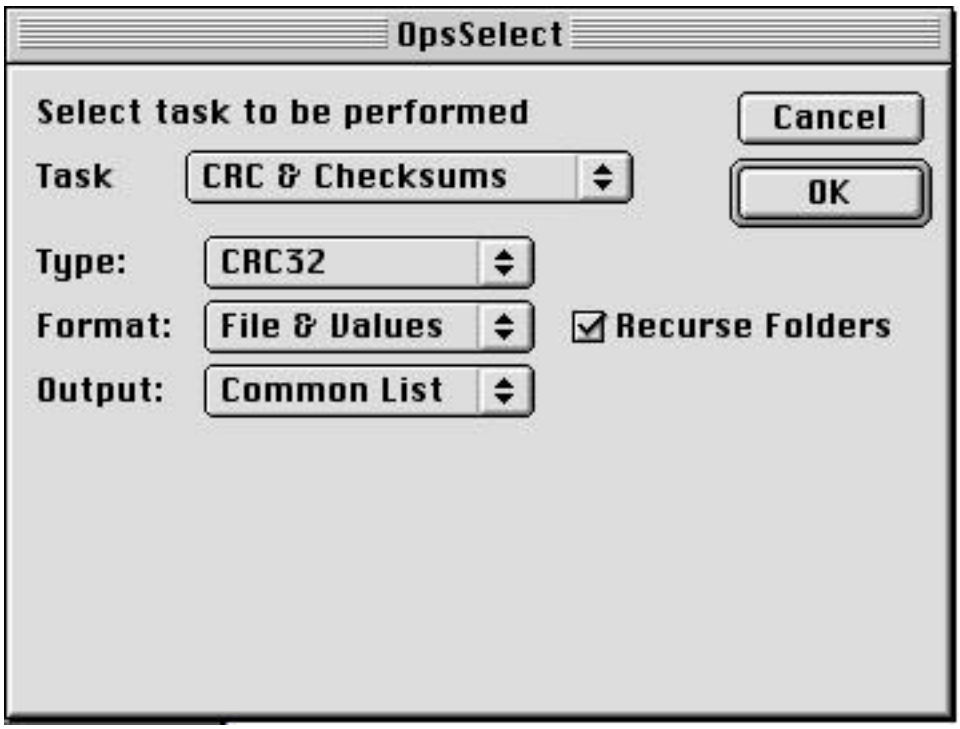
Checksum - a simple sum of all data bytes contained in a file, transmission or other object. Checksums and the other checksum like numbers defined below are used to represent the total data in that object with a single simple number. If a checksum is compared to the results obtained from a recomputed checksum for that data by the recipient, it reveals the validity of the data and whether any of it has changed since the original computation.

CRC - (Cyclic Redundancy Character) a more complex form of checksum based on any of a variety of binary polynomials and calculations based upon these polynomials. Simple checksums are extremely fast, but can easily be defeated by a variety of patterns in the data for which different data will produce the same identical checksum. CRCs are far more resistant to the effects of such data patterns but are somewhat slower due to the additional math which must be performed for each byte of data. They are still fast enough to be used in real-time data checks such as network data transmissions, mag tape unit data checks and for general data comparison use.

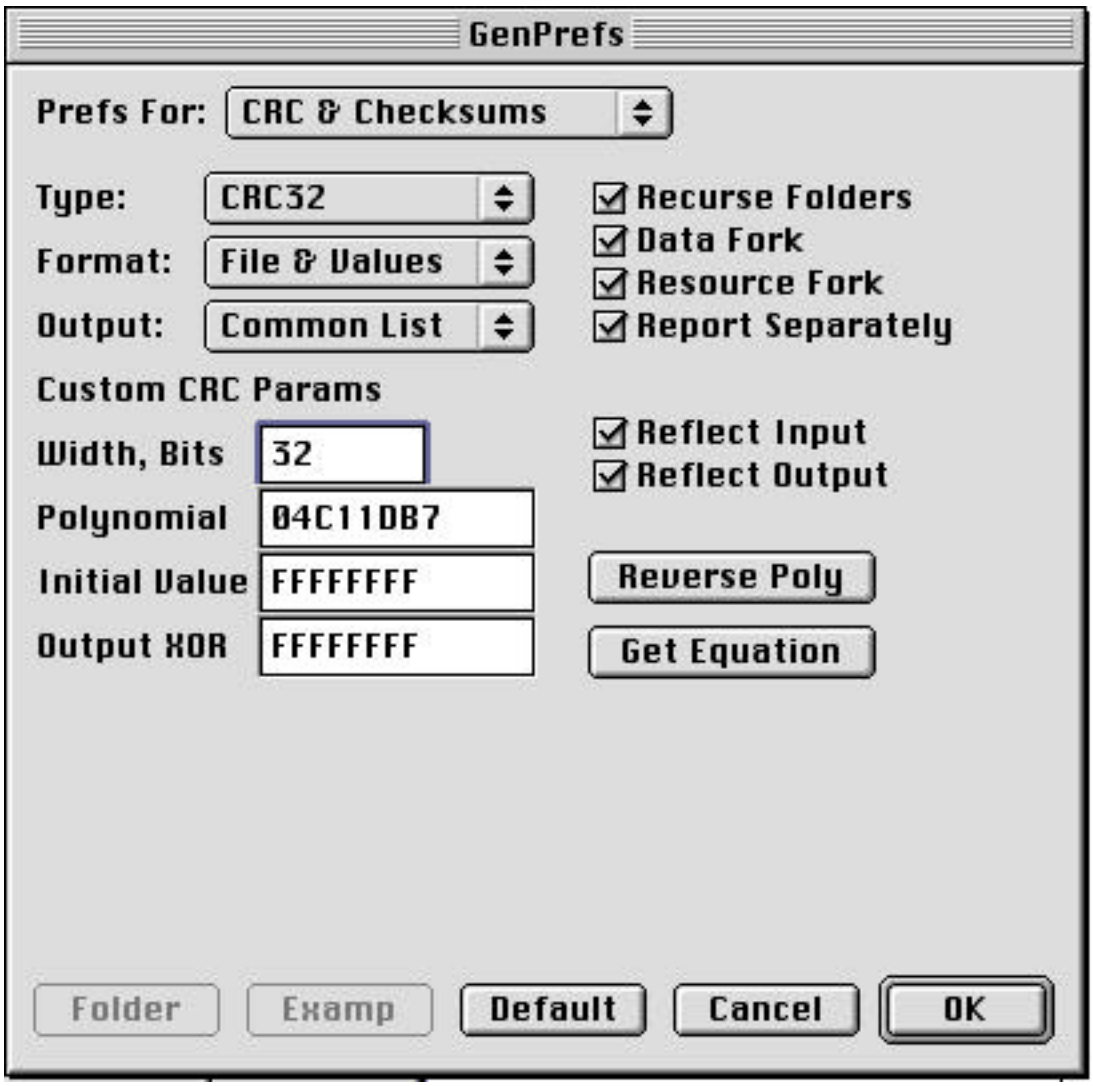
Hashing Algorithms - a hashing algorithm or hashing function is a more general term which also includes both checksums and CRCs. It refers to any operation or series of operations on a data stream which will reduce that data to a single numeric value representing that data. In common use, however, hashing functions refer to a variety of very complex but specific mathematical transformations applied to that data stream to reduce it to that representative number. They are far more resistant to the effects of data patterns but are very slow and are primarily used for cryptography and digital signatures. The two common examples included in MNT are the MD5 (Message Digest 5) and the SHA (Secure Hash Algorithm) functions. MD5 is also used in Mime encodings as a data verification method.

I would like to include a tutorial section describing the theory and practice behind CRCs, at least what I was able to learn while making the code work properly. But the truth of the matter is that CRCs and Hashing functions are almost a subject of study by themselves. Although I have a very strong background in math and moved into the software field from the field of engineering, I am not by any means a mathematician and quite frankly it took a lot of time and trial and error to get all of these functions working properly. If anyone is really interested in the details of this subject and wants to learn more, especially about the nitty gritty details on how to build and use custom CRC tables, I would suggest that you begin with the internet document "A Painless Guide to CRC Error Detection Algorithms" listed in the Credits chapter. This document served as my bible in creating the code used by MNT for many of its CRC functions.

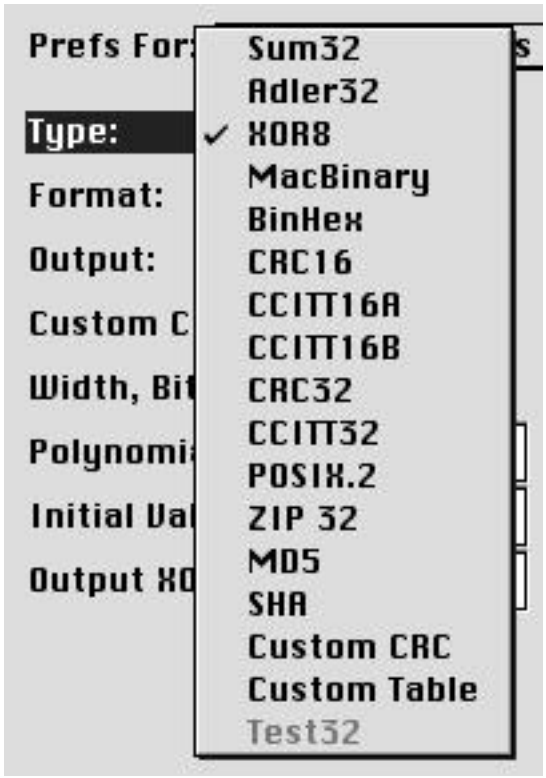
Ops Select dialog for CRC & Checksums Command



Full Prefs dialog for CRC & Checksums Command (described in full detail below)

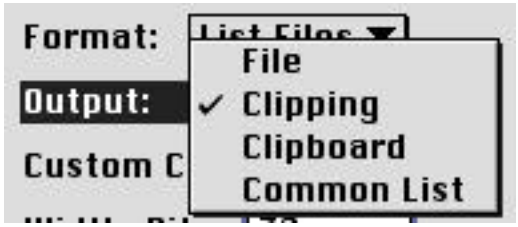


Type Popup shown in pulled down state



Format Popup shown in pulled down state

Output Popup shown in pulled down state



Type Popup - selects the particular checksum algorithm to use

Sum32	a simple sum of all bytes in each file using a 32 bit accumulator with overflow bits discarded.
Adler32	a 32 bit checksum algorithm named after its originator, Mark Adler which is reported to be much faster than the CRC32 algorithm yet still provides an extremely low probability of undetected errors. Used in some ZIP implementations.
XOR8	a simple 8 bit exclusive OR of all bytes in each file using an 8 bit accumulator.
MacBinary	a 16 bit CRC as used by MacBinary encoding.
BinHex	a 16 bit CRC as used by BinHex encoding. Uses a special CRC algorithm.
CRC16	a 16 bit CRC as used by ARC compression and others
CCITT16A	a 16 bit CRC defined by the CCITT international standards organization. A commonly used CCITT16 algorithm gives an incorrect checkfile value of $29B1. This version gives the correct check value of $E5CC. For further details see the web page at http://www.joegeluso.com/software/articles/ccitt.htm
CCITT16B	a 16 bit CRC defined by the CCITT international standards organization. This is the commonly used yet incorrect CCITT16 algorithm referenced above.
CRC32	a 32 bit CRC that is probably one of the most (if not the most) widely used CRC algorithms in existence today.
CCITT32	a 32 bit CRC defined by the CCITT international standards organization.
POSIX.2	a 32 bit CRC as defined by POSIX.2 standard. Uses a special CRC algorithm.
ZIP 32	a 32 bit CRC previously used in ZIP compression. Now replaced by CRC32.
MD5	Message Digest 5, the last of a series of hashing algorithms (MD2, MD3, MD4, MD5) that is part of RSA Data Security's RSARef crypto package. Commonly used for data validity checks in Mime encodings and many other uses. MD5 outputs an 18 byte binary digest of the data which it processes. Each 3 bytes of this digest are then transformed into 4 bytes of base64 encoded text (human readable) resulting in a 24 character text string which serves as the MD5 output.
SHA	Secure Hash Algorithm, over the course of time, holes were found in each of the MDx algorithms including MD5 so mathematicians devised a totally new algorithm which was supposed to be even tighter than MD5. This was SHA. The SHA algorithm outputs a 20 byte digest but with no commonly used method of converting this to human readable form, I have had to devise my own (in fact several). What MNT does at present is to pad the 20 byte digest with a single byte of zeros and then to base64 encode those 21 bytes into a 28 byte string in human readable form.
Custom CRC	allows the user to specify his own CRC parameters (Width, Polynomial, Initial Value, Output XOR value, Reflect In, Reflect Out) in the boxes provided and to use those parameters to generate CRC values as easily as using any of the other standard CC functions. You will typically not have any need to do this unless you know a little bit about CRCs and the math behind them.
Custom Table	allows the user to generate his own CRC lookup tables from the parameters listed above. Enter the desired CRC parameters, then just drop anything onto MNT and select the Custom Table CRC type. The custom CRC table will be output in the form selected by the Output format just as would be the case with any other CRC list. The dropped item will not be altered, but is only used as a trigger to get MNT to execute the command. All of you mathematicians out there will just have a blast with this little toy which allows you to quickly generate lookup tables for any conceivable kind of CRC function. Please try not to waste too much time playing with it though as it can be rather addictive. When these tables are used with either the normal or reverse CRC functions (as applicable) listed in the Ross N. Williams document referenced above, you will end up with a fast custom CRC generator ready to be included in your own program code. This is how at least one or two of the CRC functions used in MNT were created.

Format Popup - selects the format for the output data

File & Values — Each file's output data will consist of one line of text listing the filename, followed by length and CRC values with descriptive labels. If "Report Separately" is not checked, then data fork values will be shown before resource fork values. Two example lines follow:

Combined Forks:

SomeFile, Len = 439, CRC = $536068F6

Separate Forks:

SomeFile, DF Len = 89, CRC = $75BD909C, RF Len = 350, CRC = $9CC7C744

List Files — Each file's output data will consist of one line of text in the same format as output by the commonly used "List Files" program for the Mac. All of the many output formatting options provided by "List Files" are not supported. Only the filename, size in bytes, sum and CRC32 values are output. As is the case with the "List Files" program, only combined data and resource forks are reported. It is up to the user to also select the CRC32 type from the Type popup. An example line follows:

MacNetTools User Modes, 89 bytes, sum = $000016A9, CRC = $75BD909C

PC MPEG — Each file's output data will consist of one line of text in the same format as output by the commonly used MPEG CRC program for the PC (name currently unknown). Since this format is used for virtually all mpeg CRC lists posted in usenet, it will be supported by MNT. Each line of data consists of a filename, followed by " -- ", followed by a CRC32 value. It is up to the user to also select the CRC32 type from the Type popup. Only the data fork is reported (PC files have no resource fork). An example line follows:

DonaldQuacks.mpg.001 -- b976acc6

Output Popup - selects the type of output data

File — Outputs info to a text file in the same folder as the files/folders dropped. This file is named "xxx Output" where xxx is the name of each files/folders dropped.

Clipping — Outputs info to a text clipping in the same folder as the files/folder dropped. This clipping is named "xxx Output" where xxx is the name of each files/folders dropped.

Clipboard — Outputs info to the clipboard for subsequent paste into any other application.

Common List — Outputs info from all items dropped to a common text file in the same folder as the files/folders dropped. This file is named "CRC Output". For a single dropped item, Common List works identically to the File output option except for the name of the output file.

Recurse Subfolders - if checked, causes files within subfolders of any dropped folders to be processed in addition to the files within the dropped folder itself. Otherwise only the files in the dropped folders themselves are processed.

Data Fork - if checked, then values for each file's data fork will be reported in the output, where applicable depending on the output format chosen.

Resource Fork - if checked, then values for each file's resource fork will be reported in the output, where applicable depending on the output format chosen.

Report Separately - if checked, then values for each file's data and resource forks will be reported separately in the output. Otherwise, the combined values will be reported.

Defaults Button - resets all parameters in this panel to their original default values.

Custom CRC Parameters - for further details, see internet document referenced in Credits chapter or any good tutorial on CRCs.

Width, Bits - width in bits of the CRC polynomial and accumulator for custom CRCs or tables

Polynomial - the hexadecimal representation of the CRC polynomial to be used for custom CRCs or tables

Initial Value - initial value to which the CRC accumulator is set (hexadecimal)

Output XOR - value with which the CRC accumulator is Exclusive ORed before being output (hexadecimal)

Reflect Input - if checked, the bit order of input data will be reflected (or reversed)

Reflect Output - if checked, the bit order of output data will be reflected (or reversed)

Reverse Poly Button - causes the displayed polynomial to be reversed (including the effect of the implicit MSB). In a manner analogous to reversing logic equations using DeMorgan's rule, CRC polynomial's can also be reversed and will produce results as good as (but not identical to) the original polynomial. It is a simple but tedious process to reverse such polynomial's using pen and paper or even a text editor so this feature is provided to automate the process.

Get Equation Button - causes the CRC polynomial displayed to be converted to its binary equation form which is then copied to the clipboard for further use. The hexadecimal number displayed as the CRC polynomial actually represents the coefficients of an expanded polynomial equation where the MSB is always implied but not expressed. For example, the CRC polynomial $1021 is equivalent to:

```
-----------------------------------------------------------------
 3 3 2 2 2 2 2 2 2 2 2 2 1 1 1 1 1 1 1 1 1 1 0 0 0 0 0 0 0 0 0 0
 1 0 9 8 7 6 5 4 3 2 1 0 9 8 7 6 5 4 3 2 1 0 9 8 7 6 5 4 3 2 1 0
-----------------------------------------------------------------
|       |       |       |       |   1   |   0   |   2   |   1   |
-----------------------------------------------------------------
| | | | | | | | | | | | | |1|0|0|0|1|0|0|0|0|0|0|1|0|0|0|0|1|
-----------------------------------------------------------------
                               ^
                               ^
                          Implied MSB
```

Or in equation form as output by the Get Equation Button:

X^16 + X^12 + X^5 + X^0

CRC Parameter Table For Reference (info provided where known)

None of the following info is guaranteed except for the check values produced by each of the MNT CRC functions. The rest of the info in this table is only as correct as I believe the case to be based on my limited understanding of the information which I have read in the Ross Williams document and many others found on the internet. If any mathematicians out there wish to verify or correct me on any of the parameters listed, then I would welcome your input. I would suggest that anyone who intends to use any of the info in the following table should also read the Williams document and to decide for yourself what parameters to use.

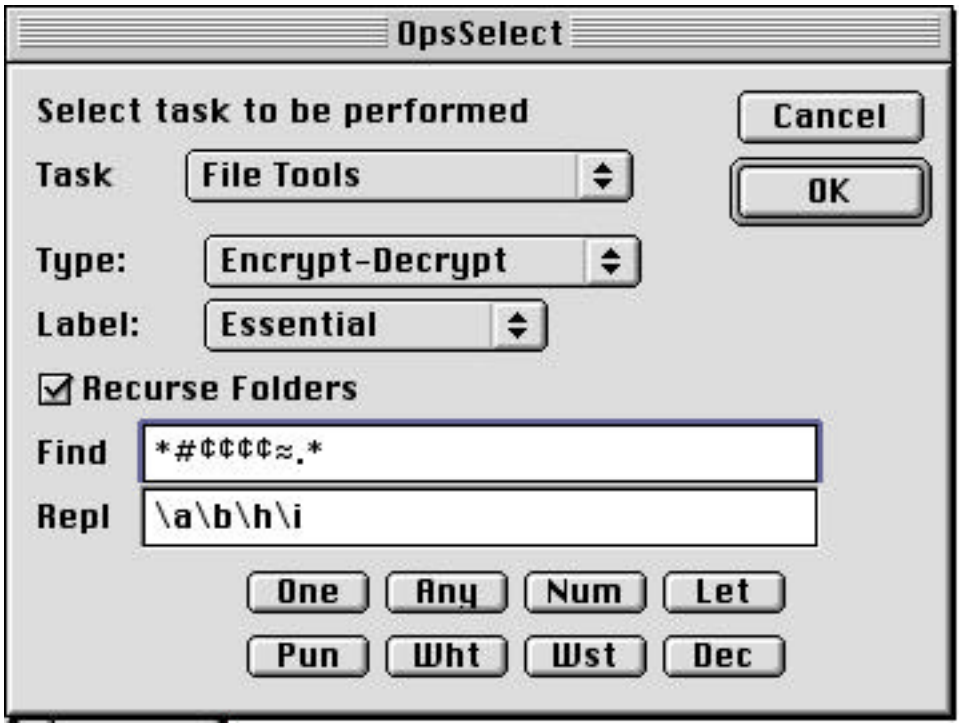
The standard method of validation of CRC functions is by running the function in question on a small CRC Check file of known contents. This file consists of 9 bytes containing the ASCII string "123456789" and nothing more (no carriage return after the 9 and no resource fork either). The values shown in the "Check Value" column below are the values obtained when each CRC function is run on such a check file. For those of you who are unaware, the $ symbol or "0x" immediately preceding a number is a designator that the number is expressed in hexadecimal form.

Type	Bits	Check Value	Polynomial	Init	XOROut	RefIIn	ReflOut
Simple Checksum Functions							
Sum32	32	$000001DD					
Adler32	32	$091501DD					
XOR8	8	$31					
CRCs							
MacBinary	16	$31C3	$1021	0	0	No	No
BinHex	16	$BEEF	$1021	0	0	?	?
CRC16	16	$BB3D	$A001	0	0	Yes	Yes
CCITT16A	16	$E5CC	$1021	$FFFF	0	No	No
CCITT16B	16	$29B1	$1021	$FFFF	0	No	No
CRC32	32	$CBF43926	$04C11DB7	$FFFFFFFF	$FFFFFFFF	Yes	Yes
CCITT32	32	$0376E6E7	$04C11DB7	$FFFFFFFF	0	No	No
POSIX.2	32	$FC9E4DC1	$04C11DB7	0 ? -1	0 ? -1	No	No
ZIP 32	32	$340BC6D9	$EDB88320	$FFFFFFFF	0	Yes	Yes
Complex Hash Functions							
MD5	18 bytes		ASCII: JfnnlDI7RTiF9RgfG2JNCw==				
SHA	20 bytes		ASCII: 98O8HYCOBHMq32eZZczDTKeuNEEA				

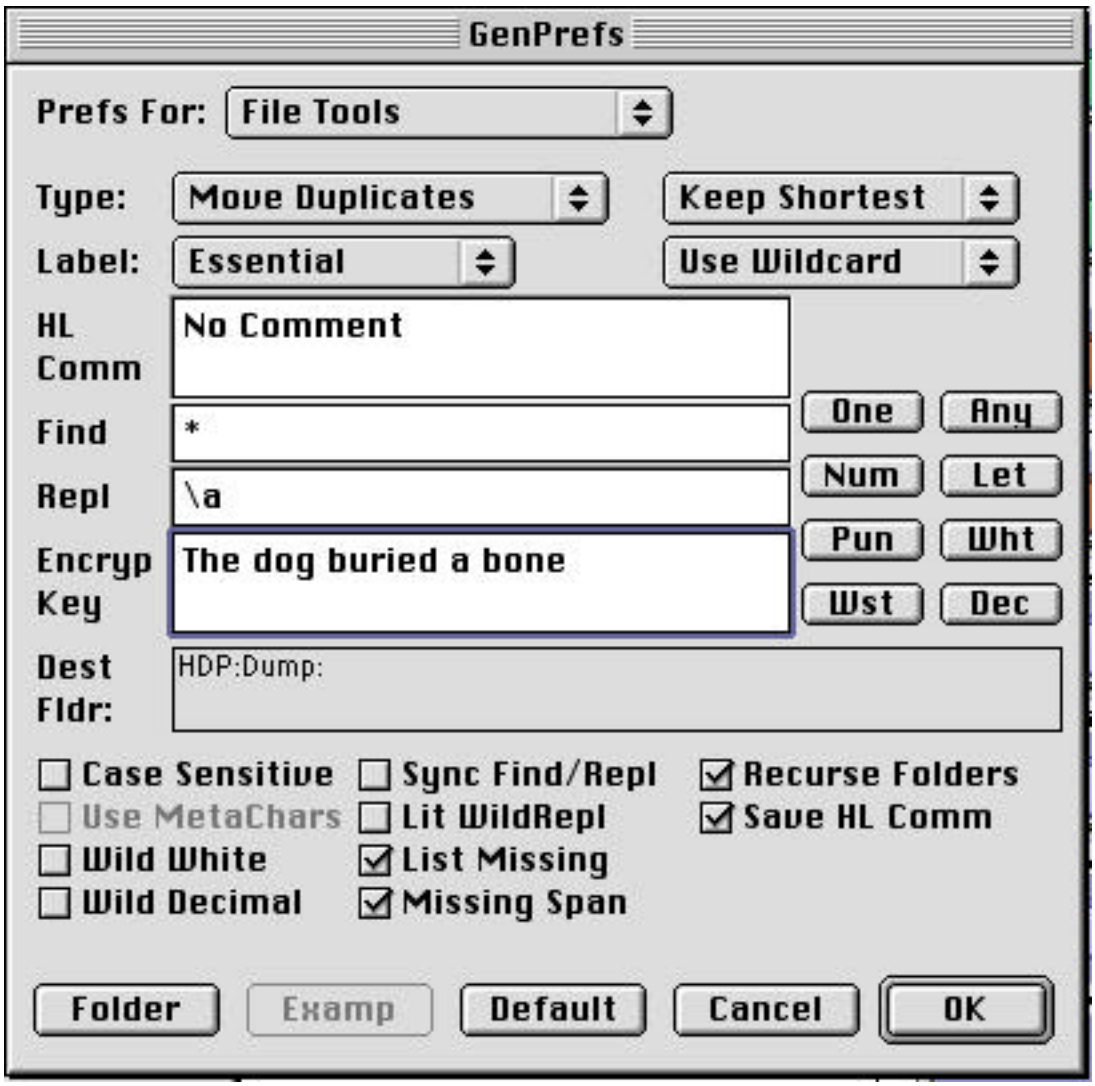
File Tools

The File Tools command is a multifunctional command which performs a variety of operations on those files or folders of files dropped onto MNT as selected by the Type popup.

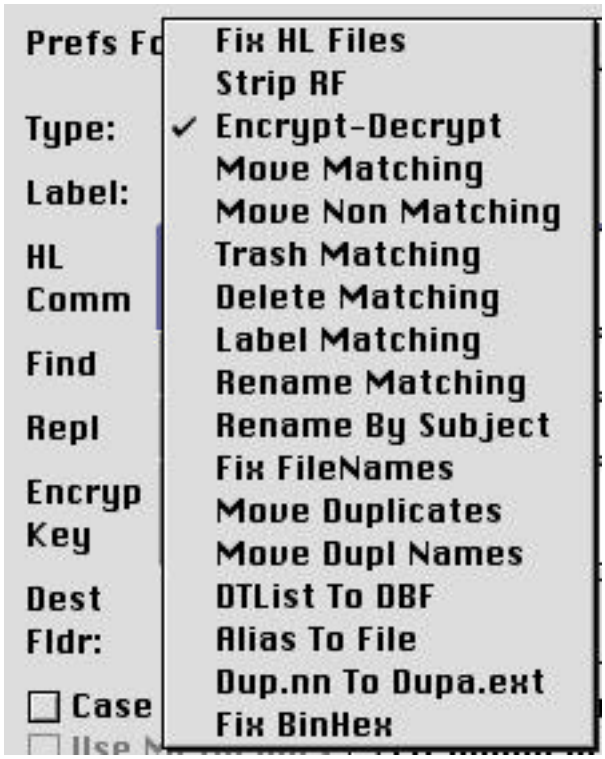
Ops Select dialog for File Tools Command



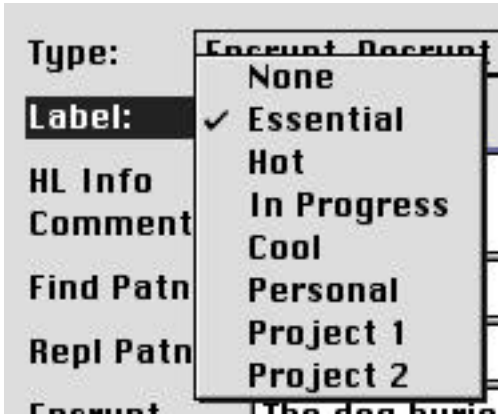
Full Prefs dialog for File Tools Command (described in full detail below)



Type Popup shown in pulled down state



Label Popup shown in pulled down state



Matcher Popup shown in pulled down state

Duplicates Preferred File Popup shown in pulled down state

Type Popup - selects the particular function to perform

Fix HL Files — Older versions of the Hotline personal networking software contain a security flaw which tags the Finder Get Info comments of downloaded files with the IP number of the server from which it was obtained. This IP number will remain on all subsequent downloads of that file and will point back to its originating source. This command replaces the existing Get Info comments of each file dropped onto MNT with the user specified text in the "HL Info Comments" box. Additionally, if "Save HL Comm" is checked and the existing comments are anything other than an IP number or domain name, then it is interpreted as valid info and will be saved to a text clipping named "xxx Info" where xxx is the original filename in the same folder as the original file.

Strip RF — Strips the Resource Fork from all files dropped onto MNT. Occasionally it is desirable to remove a file's Resource Fork, however the MacOS provides no function call to perform this operation. Once the Resource Fork is created it is there forever and cannot be removed without deleting the entire file. This command makes this operation possible simply by copying the data fork to a new temp file that does not contain a Resource Fork then deleting the original and renaming the temp file to the same name as the original. At present, Get Info comments and file dates are not preserved.

Encrypt-Decrypt — Performs a simple Exclusive OR encryption of files dropped onto MNT using the text entered in the "Encrypt Key" box as the encryption key. An Exclusive OR with a non-repeating key is generally considered to be the only form of encryption which can never be broken. For long documents, however, a single non-repeating key must be as long as the document itself which in most cases is impractical. The key used in MNT's encryption scheme has a maximum length of 255 characters and repeats when the end of the key is reached. In the rare case where the data to be encrypted is shorter than the key which the user enters, the encrypted data will be as secure as described above. In most cases, however, the data to be encrypted is much longer than the key. This results in a repeating pattern in the encrypted data itself which makes this scheme fairly easy to break. As such it is nowhere nearly as secure as PGP, DES or other highly secure methods of encryption, but it is more than enough to thwart prying eyes and to keep your mom, wife or boss from being able to see those pics or special notes that you have stored on your computer. This encryption scheme is totally symmetrical in that the same key and the same command are used both to encrypt and to decrypt the data. There is also no indication via filename or otherwise that a file is encrypted (I may add this in future versions). The user must keep track of this by himself. If you think that you have encrypted a file using MNT but are unsure, then try to open it. If it will not open then it may be encrypted. If so, then decrypt it and try again. If it opens, then you are home free. If it still will not open, then you are out of luck. If you forgot your key, then you are also out of luck - do not email me asking how to recover forgotten keys because it is not possible for me to do this. Only the Data Fork of files is affected by the Encrypt-Decrypt command.

Move Matching — Moves files dropped onto MNT to the user specified Dest Fldr (see Dest Fldr and Folder Button below) if the filename matches the specified Find Patn. See document on Pattern Matching.

Move Non-Matching — Moves files dropped onto MNT to the user specified Dest Fldr (see Dest Fldr and Folder Button below) if the filename does not match the specified Find Patn. See document on Pattern Matching.

Trash Matching — Moves files dropped onto MNT to the Trash if the filename matches the specified Find Patn. See document on Pattern Matching.

Delete Matching — Deletes files dropped onto MNT if the filename matches the specified Find Patn. See document on Pattern Matching.

Label Matching — Sets the Finder Label/Icon Color of files dropped onto MNT to the Label selected by the Label popup if the filename matches the specified Find Patn. See document on Pattern Matching.

Rename Matching — Renames files dropped onto MNT if the filename matches the specified Find Patn. The new name for the file will be based on both the Find Patn and the Repl Patn. See document on Pattern Matching. For most cases of file renaming it is often much easier to use a great little Mac utility called "A Better Finder Rename". See Credits section for URL. The MNT rename command is better suited for those cases involving complex text patterns rather than literals.

Rename By Subj — Scans the first 8192 bytes of files dropped onto MNT looking for a standard message subject line. If found, the subject line is massaged in such a way as to extract filename info where it should exist and the file is renamed to the last 28 bytes of that filename info (allowing the user 3 characters to spare). This command is useful in cases where some lame brained posters use subject lines that confuse most newsreaders in their own choice of filenames leading to totally non-descriptive filenames.

Fix FileNames — Examines filenames for suffixes (such as .00, .01 etc) following the extension if any. If such a suffix exists, the file will be renamed to the same filename without the suffix if possible. Otherwise, if such a suffix exists, and a lower numbered suffix for the same filename is not used, then the file will be renamed so as to use the lowest possible numbered suffix. This command should **NOT** be used with files which form a numbered series such as MPG segments or Stuffit segments in which such suffixes have a very specific meaning and must not be changed. Doing so would only renumber your segments into a totally meaningless hodgepodge of files. This command would be useful for renaming files derived from various decoders which have tacked on such a suffix as part of the decoding process but where the continued need for such a suffix no longer exists. Other features may be added to this command in the future such as stripping leading punctuation marks, although this can at present be accomplished through use of the Rename Matching command.

Move Duplicates — Works somewhat similarly to the List Duplicates command under List Files. Scans the files and folders of files dropped onto MNT while building a table of file info for each file. Each entry in this table is then compared to all other entries in order to detect identical files (the filename is not taken into consideration). Where duplicates are found, the first occurrence of a given file is retained and the rest are moved to the user specified Dest Fldr (see Dest Fldr and Folder Button below). As with List Duplicates, this is a very powerful tool for file comparison and sorting. It is also computationally intensive and rather slow compared to other MNT commands. NOTE that duplicate filenames are not considered in this command in any way. Likewise, the Resource Fork is not examined or taken into consideration (although this may very well become a future option). This command looks only at the actual contents and size of each file's data fork.

In order to perform this operation within an acceptable amount of time, MNT uses a simple trick to identify a file's contents. A table is built containing each file's Data Fork CRC and size values. Each entry in this table is then compared to all other entries and those files with identical values (after the first occurrence

of each) are moved. This method, while fast, is not exact. It will positively identify **ALL** true duplicates, however, there is a one in 4 billion chance that any given file's CRC value will match that of another totally different file. The file size comparison was added to further reduce this already tiny chance to the point of insignificance.

The time required for a List Duplicates operation varies depending on a number of factors including processor speed, total number of files, combined file size and to a much lesser extent, the number of Duplicates found. For example, with a 800 Mhz G4 it takes about 10 seconds to run Move Duplicates on a folder of 930 files totaling 60 MB. A list on a folder of 3145 files totaling 204 MB takes 41 seconds. Generally speaking the time required will increase geometrically with the number of files and linearly with the average file size.

Move Duplicates works similarly to the List Duplicates command under List Files with one exception. List Duplicates will list **ALL** duplicates found including the first instance of each duplicated file. Move Duplicates on the other hand will always keep the first instance of each duplicated file and will move all of the rest.

Also See Chapter on **Usage Tips**

Move Dupl Names This command works similarly to the Move Duplicates command except that files are moved based only on identical filenames and Data Fork lengths. This command is here by user request only. I do not advise its use at all unless this is specifically what you want to do because it can quite readily dump files whose content is totally unique. Move Duplicates is a far better alternative and does everything that Move Dupl Names does plus more.

DTList To DBF This highly specialized command will read an exported text file from the DiskTracker disk cataloguing program, scan that file for data in a certain expected format and output a tab-delimited summary of that info suitable for import into any database manager. Examples are provided at the end of this chapter to avoid cluttering up everything else.

Alias To File Takes an alias or folder full of aliases as its input and converts those aliases to full copies of the original file. Only the Data Fork is copied. At present, the Resource Fork is ignored. This command is typically used in conjunction with the make alias feature of the Graphic Converter image viewer for Mac. Often I have found myself in the position of sorting unorganized folders of downloaded images using the slideshow feature of Graphic Converter and found files which I wanted copy to a given folder rather than simply moving them. I was almost to the point of creating my own slideshow in MNT to implement such features but decided that this would have to wait until the next version. It is hard to beat the slideshow which Graphic Converter provides, but there are a few additional features which would be very useful to have. A move copy feature would be nice as also would be a windowed form of slideshow so that the Finder and/or other programs could be accessed without stopping the show. Anyhow, this is for the future. The Alias To File feature was added to MNT as a quick and dirty shortcut around this needed capability. Simply use the Graphic Converter slideshow while making aliases of images for which copies are desired then run Alias To File on the folder of aliases to turn them into full copies of the original images.

Dup.nn To Dupa.ext Renames files having filenames of the form DonaldQuacks.jpg.01 to the form DonaldQuacksa.jpg with the .nn suffix removed and the first available letter a-z inserted before the extension. Many times when sorting files and combining folders of files having common names such as those 1.jpg, 2.jpg, 19.jpg names favored by some unimaginative usenet posters, you will end up with a number of files having duplicate names. The move duplicates command will get rid of the

true duplicates, but the duplicate filenames will often remain with those .nn suffices that MNT attaches in such cases. While this keeps the files separate in the folders, it is more desirable to rename them without such suffixes. That is exactly what this command does.

Fix BinHex — Fixes certain BinHex encodings so that Stuffit Expander can more reliably decode them. For usenet posting of Mac binary (not to be confused with Macbinary) files where the resource fork must always be preserved, BinHex encoding is **always** the preferred method (much to the chagrin of all the yEnc yokels out there who would very much like to believe otherwise). Often the posters of BinHex encoded binaries like to add informative comments (and even poetry) to the subject lines of their posts to entertain users during long download sessions by giving them something other than filenames to read in their newsreader article listings. This is fine except for one quirk that I have noticed with Stuffit Expander. MNT will decode such BinHex encodings regardless of their subject lines, but evidently (and I wouldn't swear by this) Stuffit Expander attempts to take matters a step farther by making an additional attempt to place BinHex segments within a given message into the correct order in the unlikely event that they are misordered. In order to do so it apparently looks for and relies on the segment identifiers such as (01/22), (02/22), (21/22) etc in the article's subject line.

I noticed long ago that additional text after such segment identifiers in article subject lines would often interfere with Stuffit Expander's ability to decode such posts correctly. To get around this I would use a text editor (my own - not yet released) to perform multifile search and replace operations on such articles in order to terminate all Subject lines at the first close parenthesis which is encountered. It is somewhat cumbersome to have to do this with a text editor, so I have added this feature to MNT in order to automate the entire process. So the next time that you have some BinHex encoded usenet post that Stuffit Expander will not decode, give this little tool a try before you go asking for a repost. I have found that this is the only thing wrong with about half of all the bad BinHex posts out there.

Label Popup - selects the desired label/color to be applied to files using the "Label Matching" command.

Matcher Popup

- Use Wildcard — Text matching will be done using Wildcard matcher.
- Use Regexp — Text matching will be done using Regexp regular expression matcher.
- Use Regexs — Text matching will be done using Regexs regular expression matcher.

Duplicates Preferred File Popup - selects the preferred file to retain by the Move Duplicates and Move Dupl Names commands

- Keep First — The first file of each group of duplicate files will be retained.
- Keep Shortest — The file with the shortest filename of each group of duplicate files will be retained.
- Keep Letters — The first file with a filename not beginning with a digit or punctuation mark of each group of duplicate files will be retained. If none are found then the first file of each group will be retained.

Recurse Subfolders - if checked, causes files within subfolders of any dropped folders to be processed in addition to the files within the dropped folder itself. Otherwise only the files in the dropped folders themselves are processed.

Save HL Comm - if checked, the Fix HL Files command will save copies of all Get Info file comments which contain anything other than an IP number or domain name (or single word of any sort).

HL Info Comments - text entered here by the user will be used to replace the Get Info file comments of all files dropped onto MNT while executing the “Fix HL Files” command.

Encrypt Key - user entered encryption key for use by the Encrypt-Decrypt command.

Dest Fldr - displays the folder selected by user using the Folder button for use with any of the three Move commands. Must be located on same volume as files to be moved (at least for now).

List Missing - if checked, the DTList To DBF command will list missing file segments in the Missing Segments field (see below). If unchecked, this same command will list segments which we DO have in the Missing Segments field (that is the non-missing file segments).

Missing Span - if checked, the Missing Segments field output by the DTList To DBF command will contain spans of missing segments (ie 005-007, 011-012). If unchecked, this same field will contain a list of missing segments instead (ie 005, 006, 007, 011, 012).

Defaults Button - resets all parameters in this panel to their original default values.

Folder Button - brings up Standard File or Navigation dialog as appropriate for the user to navigate to and select the folder to be used for the “Dest Fldr”.

Buttons - One, Any, Num, Let, Pun, Wht, Wst, Dec, ExN, ExL - all are described in document on Pattern Matching

EditText fields - Find Patn, Repl Patn - all are described in document on Pattern Matching

Checkboxes - Case Sensitive, Use MetaChars, Wild White, Wild Decimal, Lit Wild Repl - all are described in document on Pattern Matching

DTList To DBF Continued from above - This is a very limited example to show the basic formats used.

Structure of Two CDs which are subsequently scanned into DiskTracker. Note that the number of the last segment in the movie (where that number is available - typically from the CRC list) is appended in parentheses to each movie's folder name. This is NOT the last segment which we have available, but rather the final segment in the movie itself if we were to be able to download it in its entirety. Note that short date format must be chosen in DiskTracker prior to export for this command to work properly.

```
ABMoov 150                    Volume / CD Name
   FLDR1                      Main Movies Folder
      Mickey3 (058)           Folder for This Movie (058 is last segment in Movie)
         Mickey3.mpg.000      Filenames for each individual segment
         Mickey3.mpg.001
         Mickey3.mpg.002
         Mickey3.mpg.003
         zMickey3.mpg(c).crc
      Donald2 (142)           Folder for Another Movie
         Donald2.mpg.000
         Donald2.mpg.001
         Donald2.mpg.002
         Donald2.mpg.003
         Donald2.mpg.004
         Donald2.mpg.005
         Donald2.mpg.006
      Minnie ()
         Minnie.mpg.001
         Minnie.mpg.002
         Minnie.mpg.003
         Minnie.mpg.004
   FLDR2                      Unused Main Folder
   FLDR3                      Unused Main Folder

ABMoov 151                    Another Volume / CD etc
   FLDR1
      Goofy1 (313)
         Goofy1.mpg.000
         Goofy1.mpg.001
         Goofy1.mpg.002
         Goofy1.mpg.003
         Goofy1.mpg.004
         Goofy1.mpg.005
         zGoofy1.mpg(c).crc
      Grumpy7 (626)
         Grumpy7.mpg.000
         Grumpy7.mpg.001
         Grumpy7.mpg.002
         Grumpy7.mpg.003
         zGrumpy7.mpg(c).crc
   FLDR2
   FLDR3
```

When the DiskTracker archive is exported to a text file, the results are similar to the following. It is this file which is subsequently dropped onto MNT to generate the summary output. Data files exported from DiskTracker can become very large, sometimes 5-10 MB of more and a full example would be too unwieldy to show here.

```
Name    Size  Kind  Created Date      Modified Date
ABMoov 150    690.4MB     CD-ROM      5/31/02     5/31/02
 FLDR1  688.8MB     folder      9/25/99     5/31/02
  Mickey3 (058)     28.6MB      folder      5/15/02     5/16/02
   Mickey3.mpg.000  12K   document    5/13/02     5/13/02
   Mickey3.mpg.001  504K  document    5/13/02     5/13/02
   Mickey3.mpg.002  504K  document    5/13/02     5/13/02
   Mickey3.mpg.003  504K  document    5/13/02     5/13/02
   zMickey3.mpg(c).crc    12K   document    5/13/02     5/13/02
  Donald2 (142)     91.2MB      folder      5/27/02     5/27/02
   Donald2.mpg.000  12K   document    5/27/02     5/27/02
   Donald2.mpg.001  660K  document    5/27/02     5/27/02
   Donald2.mpg.002  660K  document    5/27/02     5/27/02
   Donald2.mpg.003  660K  document    5/27/02     5/27/02
   Donald2.mpg.004  660K  document    5/27/02     5/27/02
   Donald2.mpg.005  660K  document    5/27/02     5/27/02
   Donald2.mpg.006  660K  document    5/27/02     5/27/02
  Minnie ()   4.2MB folder      5/16/02     5/16/02
   Minnie.mpg.001   432K  document    5/14/02     5/14/02
   Minnie.mpg.002   432K  document    5/14/02     5/14/02
   Minnie.mpg.003   432K  document    5/14/02     5/14/02
   Minnie.mpg.004   432K  document    5/14/02     5/14/02
 FLDR2  0K    folder      10/4/98     11/6/00
 FLDR3  0K    folder      10/4/98     11/6/00
ABMoov 151    682.5MB     CD-ROM      6/5/02      6/5/02
 FLDR1  680.9MB     folder      9/25/99     6/5/02
  Goofy1 (313)      161.3MB     folder      5/27/02     5/27/02
   Goofy1.mpg.000   12K   document    5/27/02     5/27/02
   Goofy1.mpg.001   528K  document    5/27/02     5/27/02
   Goofy1.mpg.002   528K  document    5/27/02     5/27/02
   Goofy1.mpg.003   528K  document    5/27/02     5/27/02
   Goofy1.mpg.004   528K  document    5/27/02     5/27/02
   Goofy1.mpg.005   528K  document    5/27/02     5/27/02
   zGoofy1.mpg(c).crc     12K   document    5/27/02     5/27/02
  Grumpy7 (626)     403.5MB     folder      5/18/02     5/27/02
   Grumpy7.mpg.000  12K   document    5/18/02     5/18/02
   Grumpy7.mpg.001  660K  document    5/18/02     5/18/02
   Grumpy7.mpg.002  660K  document    5/18/02     5/18/02
   Grumpy7.mpg.003  660K  document    5/18/02     5/18/02
   zGrumpy7.mpg(c).crc    24K   document    5/18/02     5/18/02
 FLDR2  0K    folder      10/4/98     11/6/00
 FLDR3  0K    folder      10/4/98     11/6/00
```

The summarized output obtained when the above file is dropped onto MNT is as follows. It should be obvious that this form of cataloguing saves an incredible amount of space, especially when full lists of movie segments are processed which can contain as many as 999 segments each.

Mickey3.mpg	ABMoov 150	5/13/2002	504	0	All after 003
Donald2.mpg	ABMoov 150	5/27/2002	660	142	All after 006
Minnie.mpg	ABMoov 150	5/14/2002	432	0	000, All after 004
Goofy1.mpg	ABMoov 151	5/27/2002	528	0	All after 005
Grumpy7.mpg	ABMoov 151	5/18/2002	660	626	All after 003

There should have been Last Movie Segment entries for all but Minnie - it isn't perfect but it usually does better than this. Evidently my manual fiddling with the data file to make it small enough to use for an example has had some effect on the results. This should be good enough to give you the basic idea on the use of this command though.

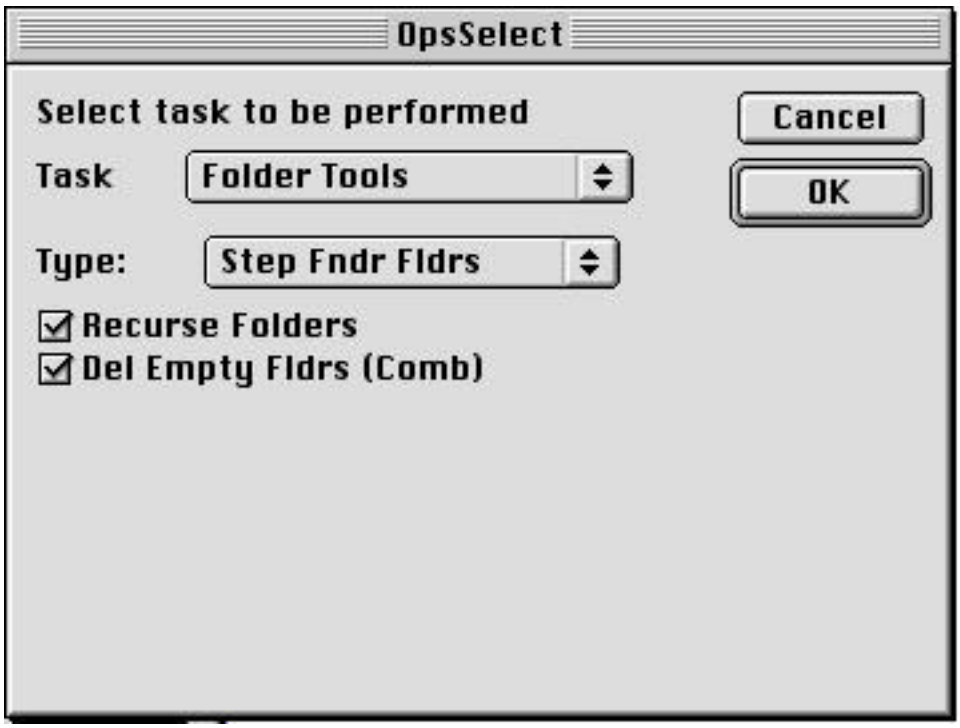
The fields are output in the following order, separated by tabs:

Movie Name
Volume/CD Name
Creation Date
Kbytes Per segment
Last Movie Segment
Missing Segments

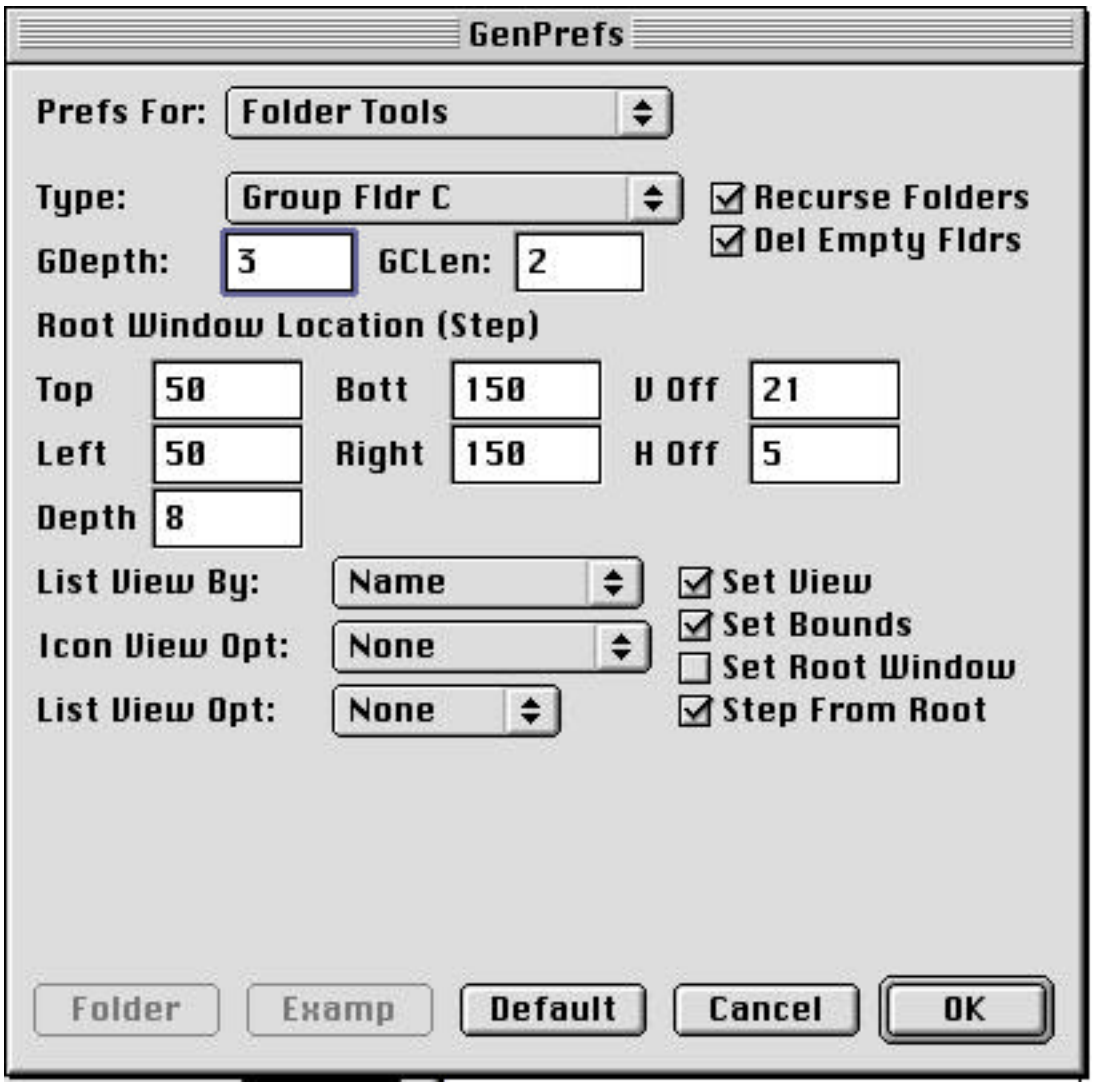
Folder Tools

The Folder Tools command is a multifunctional command which performs a variety of operations, as selected by the Type popup, on those folders (and in some cases the files which they contain) which are dropped onto MNT.

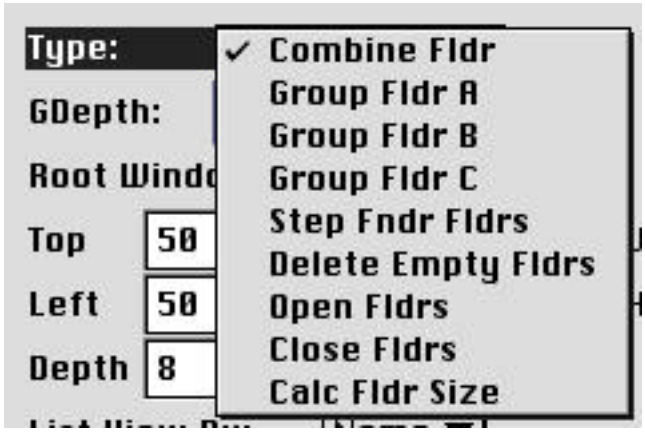
Ops Select dialog for Folder Tools Command



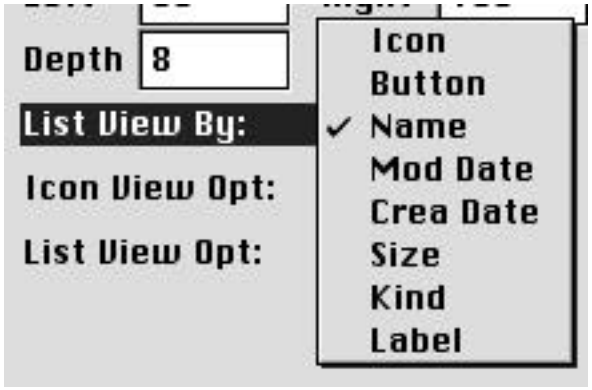
Full Prefs dialog for Folder Tools Command (described in full detail below)



Type Popup shown in pulled down state



List View By Popup shown in pulled down state



Icon View Opt Popup shown in pulled down state

List View Opt Popup shown in pulled down state

Type Popup

Combine Folder - Combines the contents of a given folder with all the contents of its subfolders, adding an index (.00, .01 etc) to any filename which will be duplicated in the process. If “Del Empty Fldrs” is checked, all empty folders left behind by this operation are deleted. “Recurse Folders” must be checked for this command to do anything.

Group Fldr A

Groups files in a given dropped folder into subfolders based on shared filename text characteristics. For example if we have a folder containing the following files, they will be grouped as shown below. If a folder already exists in the dropped folder matching the characteristics of a given file then the file will be moved into that folder and a new folder will not be created. Each of the three "Group Fldr" commands work similarly, but use a different algorithm to analyze filename text for common characteristics. The value entered for GDepth specifies the minimum number of files, each having the same common text, that must be found for a folder to be created for that group. A value of 3 to 5 (or more) are appropriate depending on user preferences.

"Group Fldr A" scans each filename left to right looking for the first '.' character indicating the beginning of a file extension, then from that point scans backwards looking for the first non-digit character following the first sequence of digits. What remains is the folder name which is used. This method, while fast, assumes a fairly rigid filename format and is best left unused as a relic from MNT's past.

MyFldr:
- Cat01.jpg
- Cat02.jpg
- Cat03.jpg
- Dog01.jpg
- Dog02.jpg
- Dog03.jpg
- Dog04.jpg
- Goat01.jpg
- Goat02.jpg

Grouped Folder Hierarchy after Group Fldr:

MyFldr:
- Cat:
  - Cat01.jpg
  - Cat02.jpg
  - Cat03.jpg
- Dog:
  - Dog01.jpg
  - Dog02.jpg
  - Dog03.jpg
  - Dog04.jpg
- Goat01.jpg
- Goat02.jpg

Group Fldr B

(See Group Fldr A). Groups files in a given dropped folder into subfolders based on shared filename text characteristics. Group Fldr B works identically to Group Fldr A and uses the same folder name determination algorithm as described above. the difference between the two is that Group Fldr B uses a separate somewhat more sophisticated function to extract the filename content from the filename with extension included. Both make an attempt to second guess how files will be named and as a consequence can sometimes give unusual results.

Group Fldr C (See Group Fldr A). Groups files in a given dropped folder into subfolders based on shared filename text characteristics. Group Fldr C is much better than the other two Group Fldr commands and uses an entirely different, although slower and more complex, method of extracting folder names from filename text. The algorithm used here looks ahead of the current file and scans the names of multiple files at once, picking out the common text among them all in a left to right manner. This common text is then examined to remove artifacts such as common trailing digits or punctuation marks an what remains is then used for the folder name for those files matching that name.

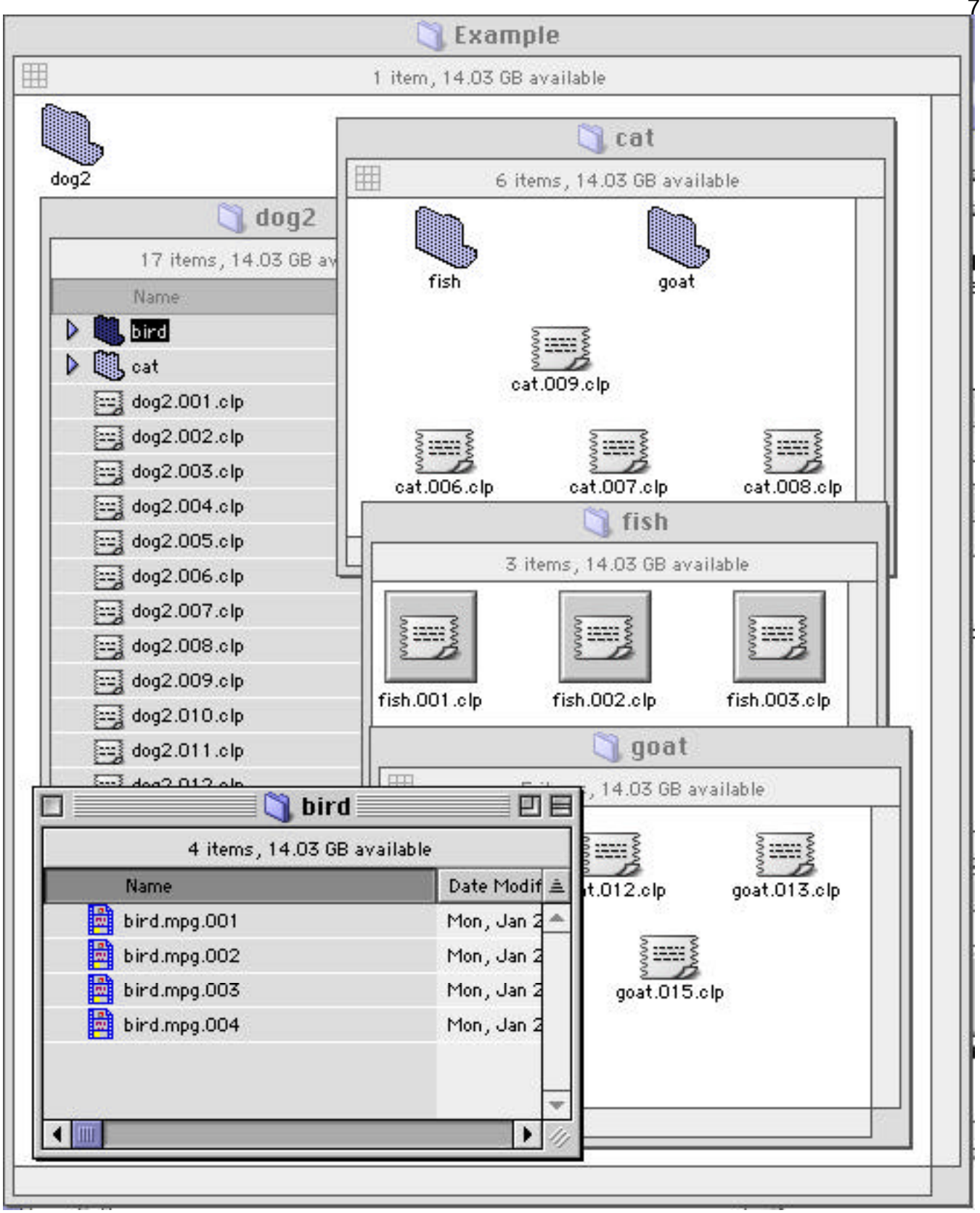
Group Fldr C does have a nasty little habit of trying to create common folders for files having the same first letter (or number) and if left to itself will tend to create folders with names such as a, b, c etc which will override longer desired folder names that are subsets of the shorter names. in order to circumvent this quirk, Group Fldr C has one additional parameter used to control it. The value entered for GCLen determines the minimum length of common text which will be allowed to be used as a folder name. Typically a value of 2 (or more) is sufficient to prevent the behavior described above.

Step Fndr Fldrs Organizes a folder hierarchy so that the Finder folder window view, location and size are consistent with each other throughout the hierarchy. Anyone who has ever dealt with a folder which has many subfolders (and perhaps subfolders of those subfolders) knows that it takes time and effort to organize these in a usable manner so that they do not end up creating a random mess on the desktop. The Step Fndr Fldrs command automates this process for you to a large extent and is especially useful for dealing with folders in List View. It does this by setting the view type of each of the subfolder Finder windows to the view specified by the "List View By" Popup (List View is a misnomer) and by setting the window's location and size to a value dependent on a number of settings. These will be described independently under each of the control descriptions below rather than trying to present all of the setting combinations at once. Note that in these descriptions, the abbreviation SFF will be used to represent Step Fndr Fldrs. Since there are no MacOS calls that directly (and quietly) affect these settings which the Finder owns, this command must be implemented through a series of AppleEvents to request that the Finder change these settings for us. AppleEvents are very slow so you will be able to watch as the Finder makes these changes one after another after another. The first time that you see Step Fndr Fldrs running, it will look like a case of window twiddling gone mad as windows start flying around your screen, opening, changing view, moving, resizing and closing and generally looking like your computer has gone totally out of control. The results will please you however, when you find that with the default settings, each level of subfolders neatly line themselves up in a stepped staircase manner with respect to the dropped folder.

For example we begin with a folder hierarchy as shown by the tree list below:

```
Example:
   dog2:
      dog2.001.clp
      dog2.002.clp
      dog2.003.clp
      dog2.004.clp
      dog2.005.clp
      dog2.006.clp
      dog2.007.clp
      dog2.008.clp
      dog2.009.clp
      dog2.010.clp
      dog2.011.clp
      dog2.012.clp
      dog2.013.clp
      dog2.014.clp
      dog2.015.clp
      bird:
         bird.mpg.001
         bird.mpg.002
         bird.mpg.003
         bird.mpg.004
      cat:
         cat.006.clp
         cat.007.clp
         cat.008.clp
         cat.009.clp
         fish:
            fish.001.clp
            fish.002.clp
            fish.003.clp
         goat:
            goat.011.clp
            goat.012.clp
            goat.013.clp
            goat.014.clp
            goat.015.clp
```

The Finder organization of these folders might be as follows. Sometimes this is exactly what you want if for instance, you wanted to be able to see all of the folders at once. However, with really large numbers of files and folders any attempt at such an organization would certainly take up every square inch of your desktop space while begging for more. If the same or repetitive actions are to be taken on each folder, then it is far more convenient to have the windows of the same size and stepped from their root folder with an offset that is dependent on the folder depth in the hierarchy.


Example
1 item, 14.03 GB available
dog2
cat
6 items, 14.03 GB available
fish
goat
cat.009.clp
cat.006.clp
cat.007.clp
cat.008.clp
dog2
17 items, 14.03 GB av
Name
bird
cat
dog2.001.clp
dog2.002.clp
dog2.003.clp
dog2.004.clp
dog2.005.clp
dog2.006.clp
dog2.007.clp
dog2.008.clp
dog2.009.clp
dog2.010.clp
dog2.011.clp
fish
3 items, 14.03 GB available
fish.001.clp
fish.002.clp
fish.003.clp
goat
goat.013.clp
goat.015.clp
bird
4 items, 14.03 GB available
Name
Date Modif
bird.mpg.001
bird.mpg.002
bird.mpg.003
bird.mpg.004
Mon, Jan 2

After running Step Fndr Fldrs on the “dog2” folder, the following is what we get with each folder’s view type and size set to the view type and size of the root folder (dog2) while its position is offset from dog2 by an amount dependent on its depth. Note that some of the folders are not visible since they are placed directly behind others. In particular, the bird folder is located directly behind the cat folder and the fish folder is located directly behind the goat folder. This is only one of the folder organization methods available with Step Fndr Fldrs, but is the default (and most useful) case.

Delete Empty Fldrs — Deletes all empty folders within a folder dropped onto MNT. Works the same as the "Del Empty Fldrs" option of Combine Folder but is available for use independently of that command. "Recurse Folders" must be checked for this command to do anything.

Open Fldrs — Opens the folder dropped onto MNT. If "Recurse Folders" is checked, then all subfolders and subfolders of those subfolders will also be opened. This is a user requested command.

Close Fldrs — Closes the folder dropped onto MNT. If "Recurse Folders" is checked, then all subfolders and subfolders of those subfolders will also be closed. This is a user requested command.

Calc Fldr Size — Does nothing at present other than to open and immediately close the folder dropped onto MNT. This command is still under construction.

SFF will be used as an abbreviation for Step Fndr Fldrs in all descriptions below:

List View By Popup — Used by SFF command to select the Finder window view to be set for each folder affected by this command. The selections are identical to those available under the Finder's View Menu Arrange / Sort List items with Icon and Button added. The name which I have chosen for the popup title is a misnomer since it is applicable regardless of whether the view is a List View or not. A popup title of "Finder View" would be more appropriate.

Icon
Button
Name
Mod Date
Crea Date
Size
Kind
Label

Icon View Opt Popup — Used by SFF command to select the Options available under Icon View mode. This selection is currently unused but may be implemented in the future.

None
Grid Snap
Keep Arranged

List View Opt Popup — Used by SFF command to select the Options available under List View mode. This selection is currently unused but may be implemented in the future.

None
None

GDepth — Used by all Group Fldr commands to determine the number of files having common text characteristics required in order for grouping to be triggered. Values of 3 to 5 (or more) are appropriate depending on user preferences.

GCLen — Used by Group Fldr C command to determine the minimum length of common filename text which will be allowed as a folder name. Typically a value of 2 (or more) is sufficient to prevent the behavior described under the "Group Fldr C" command description.

Top, Left, Bottom, Right — Used by SFF command, but only if "Set Root Window" is checked. The values entered in these four boxes determine the screen coordinates in pixels to which the root window dropped onto MNT is set by the SFF command. Window positions and sizes of all subfolders are set with respect to these initial coordinates.

For those of you who are unaware, screen coordinates on the Mac are measured from the top left corner of the screen with values increasing in the rightward and downward directions.

V Off, H Off — Used by SFF command but only if "Step From Root" is checked. Specifies the vertical and horizontal offsets by which each folder window will be positioned with respect to the window of its parent folder. For MacOS 8 and 9 standard Finder windows, the distance from the top left corner of the window to the top left corner of the windows content region is 21 pixels vertically and 5 pixels horizontally (the default values for V Off and H Off respectively). This results in the nice stepped effect in the example above but the user is free to specify any offsets which please him.

Depth — Used by SFF command. Specifies maximum folder depth relative to the dropped folder to which MNT will perform the SFF operation. Since this command must use AppleEvents to do its work, it is very slow and if the user should accidentally drop a folder having many deeply nested subfolders with no limit of the depth of this command then that user might be waiting for a very long time for the command to complete.

Set View — Used by SFF command. If checked, MNT will set the Finder Window View as specified for all folders which SFF processes.

Set Bounds — Used by SFF command. If checked, MNT will set the Finder Window Bounds (position and size) as specified for all folders which SFF processes.

Set Root Window — Used by SFF command. If checked, MNT will set the root window to the values entered in the "Top, Left, Bottom, Right" boxes. Otherwise the root window's existing coordinates will be read by MNT to be used as a reference point for the offset locations of the windows of all subfolders processed.

Step From Root — Used by SFF command. If checked, MNT will step subfolder window positions with respect to the position of the root window and having offsets as described under the description of the "V Off, H Off" items above. Otherwise no offsets will be applied and all windows will be piled up on top of each other.

Recurse Subfolders — if checked, causes files within subfolders of any dropped folders to be processed in addition to the files within the dropped folder itself. Otherwise only the files in the dropped folders themselves are processed.

Del Empty Fldrs — Used only with the Combine Fldr command. If checked, the empty folders resulting from the Combine Fldr command will be deleted.

Defaults Button — resets all parameters in this panel to their original default values.

Miscellaneous Utilities

See section on Utility Menu.

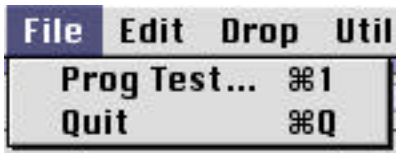
MacNetTools Menus

Apple Menu

MNT's Apple Menu has only two items.

About MacNetTools...	Displays the MacNetTools About Box.
Read License...	Displays the MacNetTools License Agreement.

File Menu



Prog Test...	Used only for testing purposes. Menu item is disabled on released version.
Quit	Quits MacNetTools.

Edit Menu

Cut	Performs Cut operation where applicable
Copy	Performs Copy operation where applicable
Paste	Performs Paste operation where applicable
Clear	Performs Clear operation where applicable
Select All	Selects All text where applicable

Manual Select — Allows user to manually select the MNT command to be executed followed by the file or folder upon which this command is to be run as opposed to dropping those files onto the MNT program icon. This is a remnant of earlier versions of MNT which will probably seldom be used.

Edit FT List... — Brings up the File Type Editor allowing the user to edit the SFT (Set File Type) list of filetypes used by the Set FType command. See chapter on File Type Editor below.

Edit Adj List... — Brings up the File Type Editor allowing the user to edit the AFT (Adjust File Type) list of filetypes used by the Adj All FType command and the built in filetyping function. See chapter on File Type Editor below.

Load Default Lists — Resets both the SFT and AFT lists to their default sets. Note that if this command is executed, all user changes and additions to both lists will be permanently lost and must be reentered. If the Option key is held down while this command is selected from the menu, then the AFT list will be set to my own personal preferred filetypes (which many users might consider to be somewhat ancient and outdated).

For those savvy users who are highly experienced with the use of ResEdit or Resorcerer, use of the following resource IDs will allow you to create your own default and preferred SFT and AFT lists so that you never do lose any of your changes and additions.

The original default lists are stored in the MacNetTools application file with the following resource IDs:

SFT List	Res Type ‘CREA’	Res ID 148
AFT List	Res Type ‘CREA’	Res ID 149
My Preferred AFT List	Res Type ‘CREA’	Res ID 150

The lists as used and edited by the user are stored in the MacNetTools Prefs file (inside of boot System Folder:Preferences) with the following resource Ids:

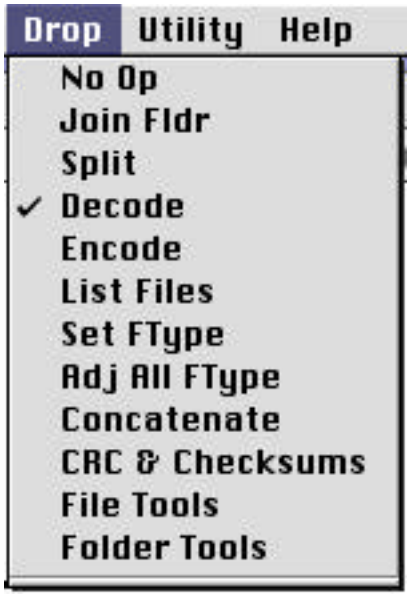
Active SFT List	Res Type ‘CREA’	Res ID 128
Active AFT List	Res Type ‘CREA’	Res ID 129

The format of the two is otherwise identical. In order to retain your own edited and otherwise modified lists, simply copy the lists in the Prefs file to the MacNetTools application file (as always working on a copy would be best), then renumber them to the appropriate Ids. If you don’t understand this description, then you probably should not be attempting such editing tricks.

Show/Hide Status Window — Shows or Hides the Status Window as applicable. See chapter on Status Window below for more information.

App Preferences... — Brings up the main preferences dialog for modification independently of any commands.

Drop Menu



The Drop Menu selects the operation which will take place if a file, folder or alias is dropped onto MacNetTools, but only if the input source is "Get Command from Drop Menu" (see "Getting Started" chapter). The available choices are as illustrated above.

Utility Menu

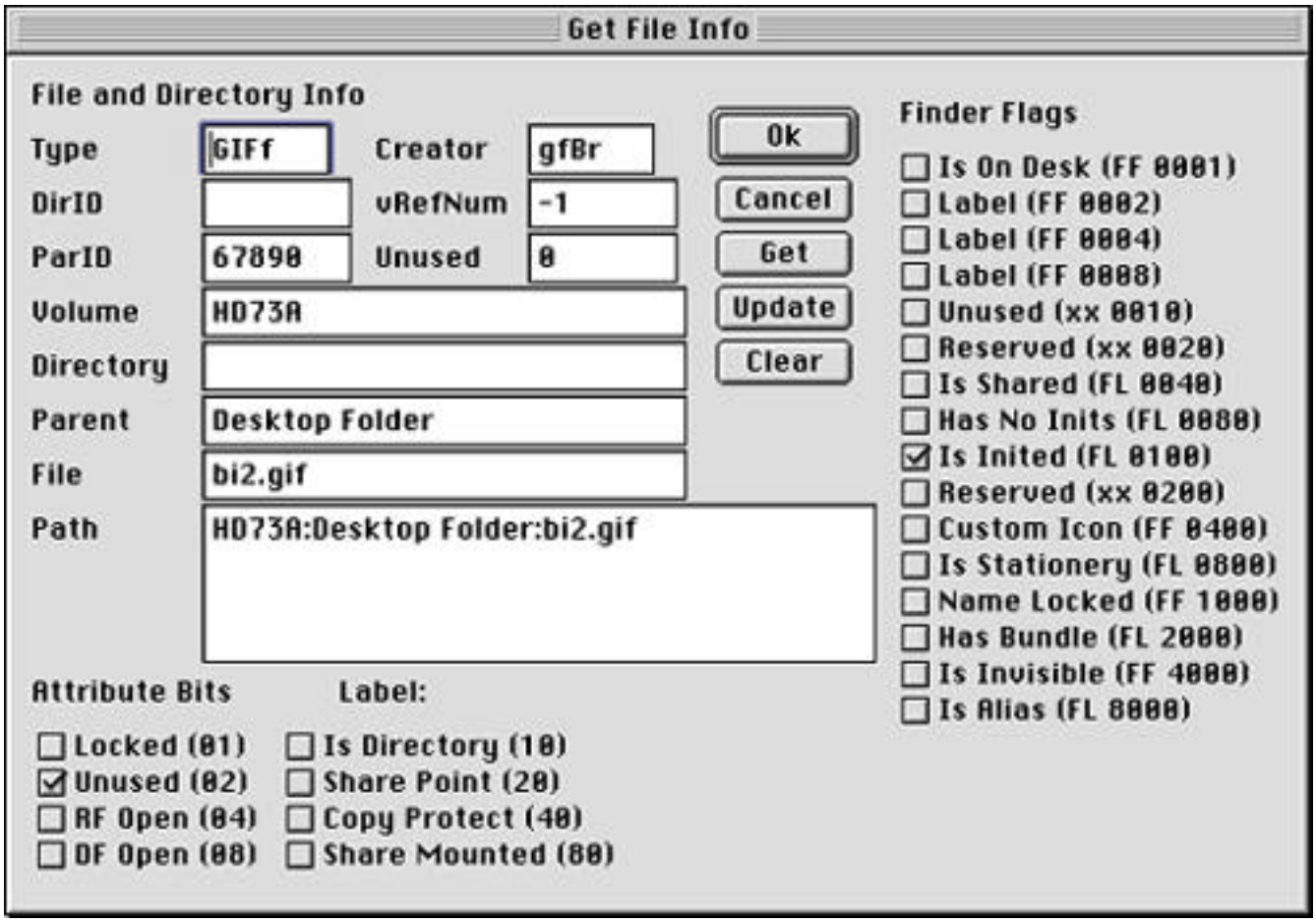
The Utility menu contains miscellaneous commands which do not fit elsewhere or which cannot be executed under a secondary thread as are all other MNT commands. Although the dialog gives the impression that it may be used to change the information for a file, this command does not at present support the setting of any data. It is used exclusively as an information display tool. Furthermore, since this command is intended primarily as a tool for programmers who already know the meanings of most of these terms, there will be no in depth explanation of the meanings of each of the items displayed. For full info on this see the File Manager chapter of Inside Macintosh.

Get/Set File Info...

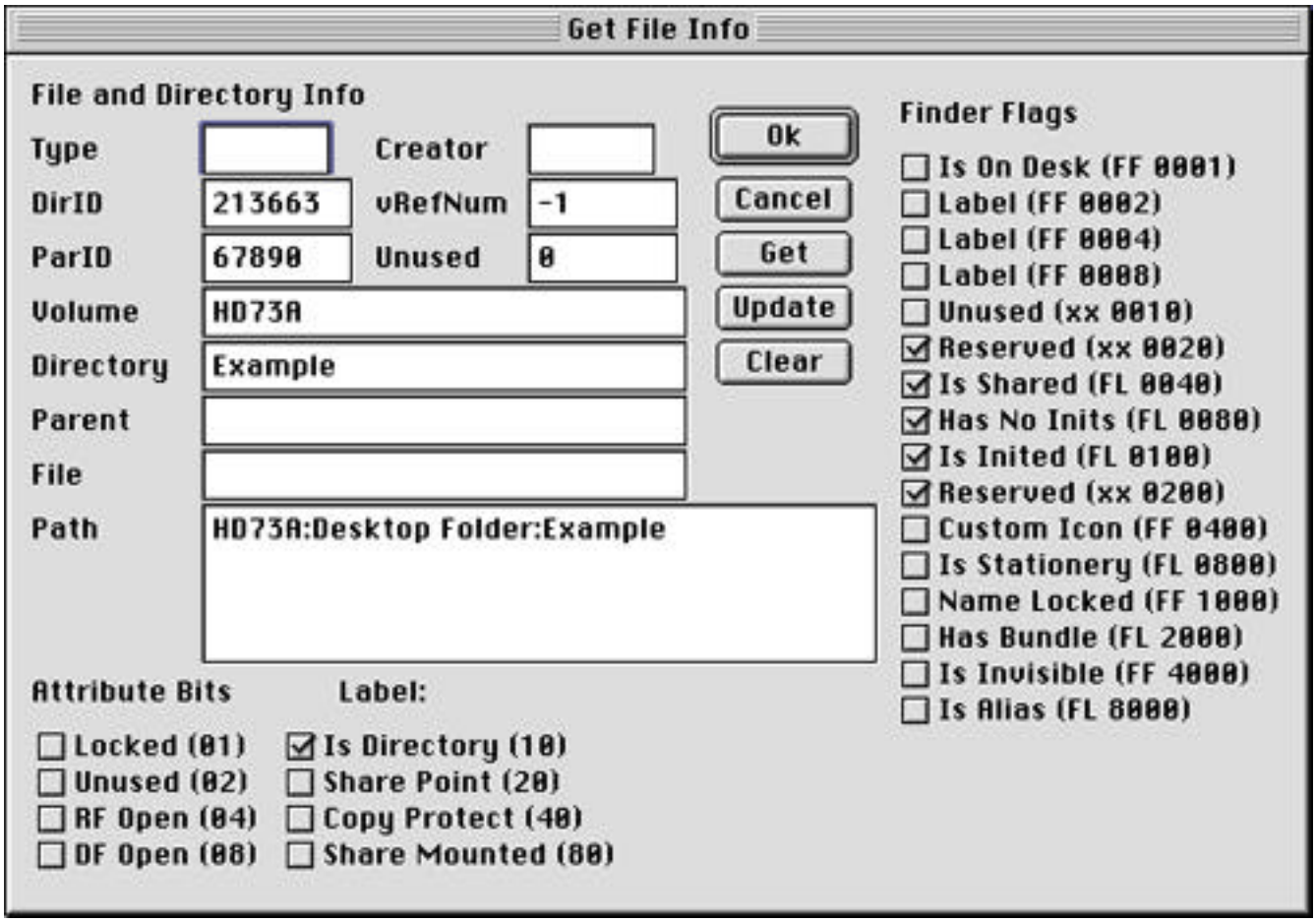
Presents a dialog which allows the user to obtain file system information on any file or folder. Contrary to the command name, it does not currently set any file info (possible future feature).

OK Button	Dismisses dialog. Does not alter file data.
Cancel Button	Dismisses dialog.
Get Button	Brings up a Standard File / Navigation dialog allowing the user to browse for and select a file or folder whose information is to be shown. This information is then displayed in the applicable boxes and checkboxes.
Update Button	Updates the flag bits for any changes which have manually been made to the file selection. Does not update the file for changes made to the flag bits. This is rather ambiguous and this button really needs a different name.
Clear Button	Clears all information displayed.

Get File Info Display for a File (reduced in size to fit page)



Get File Info Display for a Folder (reduced in size to fit page)



Wildcard Match Check

Moved to separate document on Pattern Matching

Step Fndr Fldrs

Invokes the same function as available via the Folder Tools command for dropped folders. The dropped (and threaded) form of this command was originally presenting problems under the Carbon environment and this menu item was an attempt to determine if AppleEvents behaved somehow differently under Carbon as opposed to the original PPC version of this command. The problem was resolved and this menu version of the command should probably be removed from MNT, but I am not quite willing to trust AppleEvents so completely yet.

Open Fldrs In Trash

By user request, this command opens all folders in the trash can. If the Option Key is held down while selecting this command, then all subfolders of these folders will also be opened, otherwise only the top level folders will be opened. The reasoning behind the command is that when moving a lot of files and folders around, it is possible to slip up and mistakenly trash a folder that the user thinks is empty, but which really contains wanted items. By opening up all those folders before emptying the trash, it is possible to see what is being deleted before it happens.

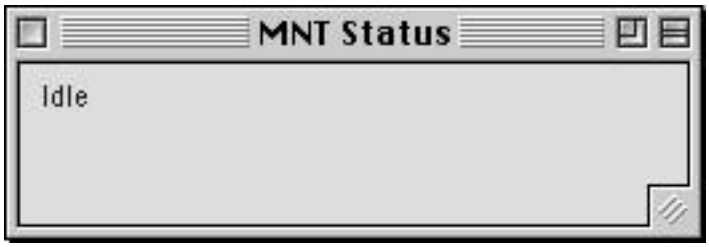
Close Fldrs In Trash

By user request, this command closes all open folders in the trash can. If the Option Key is held down while selecting this command, then all subfolders of these folders will also be closed, otherwise only the top level folders will be closed. This is the reverse of the “Open Fldrs In Trash” command and the quick way to get all those open folders closed... of course just emptying the trash like you set out to do in the beginning will accomplish the same thing too.

The Status Window

The Status Window provides continuous progress info and user feedback on whatever MNT is doing at any given time. It may take several different forms. The Status Window is displayed by selecting the “Show Status Window” item on the Edit menu which then becomes the “Hide Status Window” item. To hide it, simply select the “Hide Status Window” item which then becomes the “Show Status Window” item.

Status Window with MNT at Idle



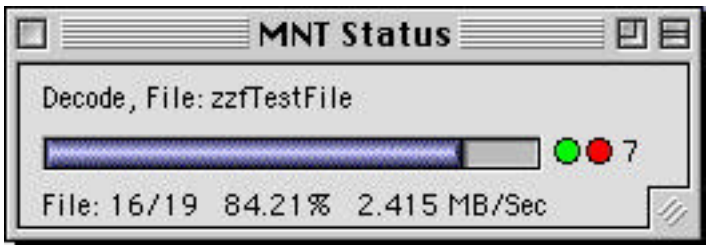
Normal Status Window with MNT running

The normal MNT Status Window displays six items.

1) Description of operation and file being processed
2) Progress bar control
3) Numeric progress stats and average input processing speed (varies widely depending on command being executed, machine and data).

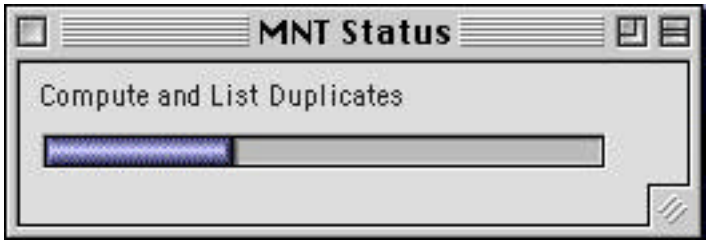
   Beginning with version 1.2.1 three useful little gizmos have been added to the right of the progress indicator to serve as a visual indication to the user of errors. These error indicators are not disabled by the Log Errors control in the General App Prefs but are disabled by the individual error disable controls such as in the Decode Prefs. They are (from left to right):

4) An “LED” indicator showing the status of the most recently completed operation, green for no error and red for error.
5) Another “LED” indicator showing if any errors have been detected since the Status window was last in its Idle state (and it goes idle momentarily between each item dropped), green for no error and red for error. This indicator is not displayed until errors occur.
6) A numeric count of the errors that have been detected since the Status window was last in its Idle state. There is room for 3 digits at the minimum Status window size. This count is not displayed until errors occur.



The command defined MNT Status Windows display specific progress information that is applicable only to certain commands. There are only a few of these and they contain only two items. Two examples follow.

1) Command specific descriptive info
2) Progress bar control



Automatic File Typing

MNT has a built in filetyping function similar to but independent of Internet Config's File Mapping feature. Most of the files generated by MNT are passed through this function which automatically determines what file type and creator to assign to a given file based on its extension if any is present. The only exceptions are those cases where the filetyping info is included with an encoded file itself such as with MacBinary and BinHex or Mime encodings that contain Mac filetype info. All other files are typed by the built in filetyping function. For those who are unenlightened about Mac file type and creator codes, these codes are used by the Finder to determine which icon to assign to a given file and what application to use to open them with when they are double clicked.

In order to facilitate this filetyping function, MNT maintains two separate filetype lists which I call the SFT list (for Set File Type) and AFT list (for Adjust File Type). The SFT list is the full list loaded from Internet Config plus any additional items which the user adds. It is used exclusively for the user to select types for the Set File Type command. Since its usage is not speed dependent, it may contain a very large number of file types. The AFT list is a much smaller filetype list which is used for both the Adjust File Type command and for the built in filetyping function. The AFT list, although unlimited in size, should contain no more than 64 entries. If more than 64 entries are provided, only the first 64 will be used by MNT. At program startup, this list is read and reprocessed into a form which although it occupies more memory, is much more rapidly accessed than the raw lists themselves. It is this information that is used on the fly to determine the filetypes to be used for decoded files and mime types to be used for mime encoded files based on the filename extensions. Both lists are fully editable by the user.

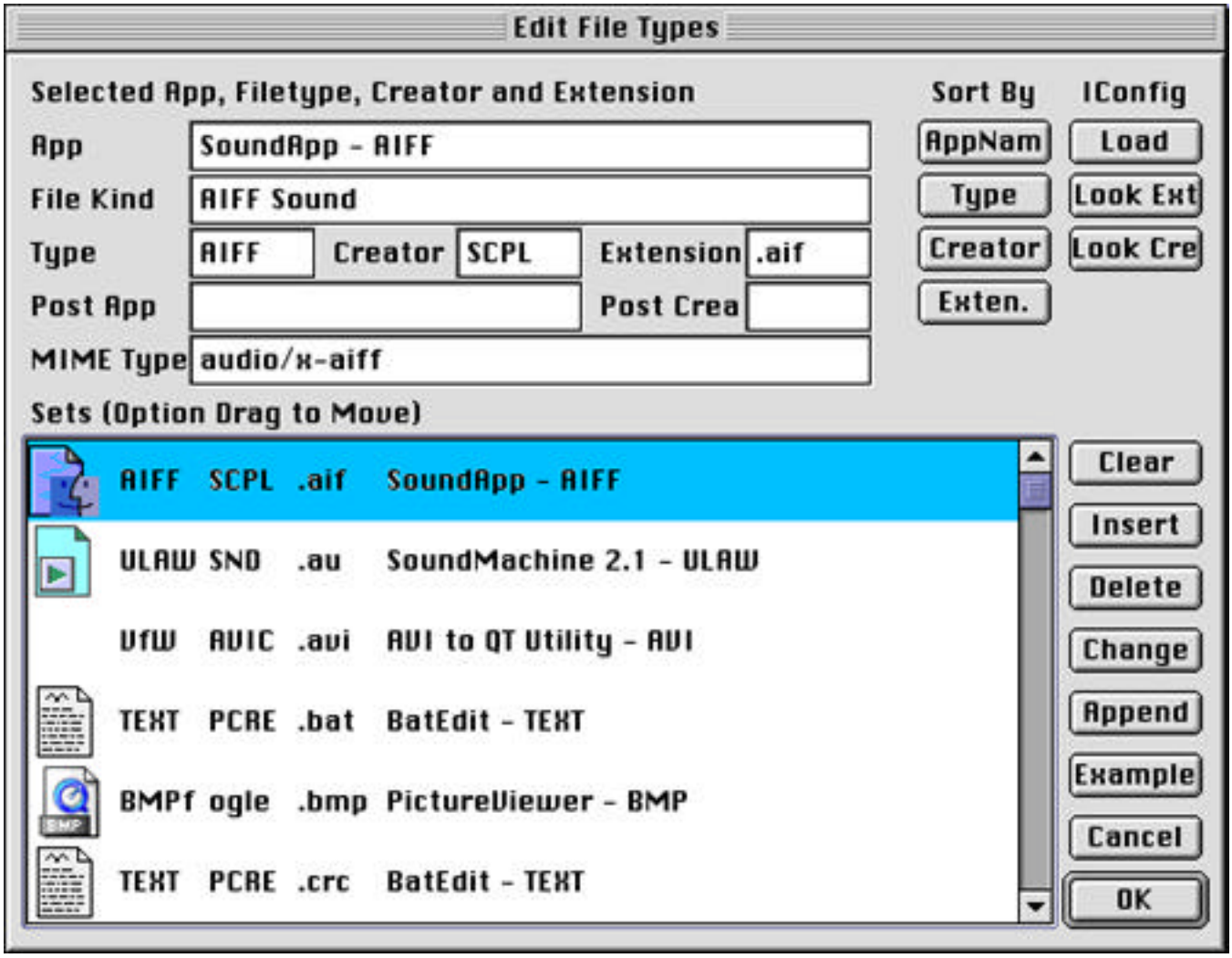
The File Type Editor

The file type editor is a means by which the user can associate file extensions with specific system filetypes, the applications which created them, their associated desktop icons and specific mime types. Much of this information is the same as that contained in the Internet Config utility (which you should get and install - its free), but it is handled in a slightly different manner for added speed. The File Type Editor is invoked by either of the two following menu items on the Edit menu as described above.

Edit FT List... Brings up the File Type Editor allowing the user to edit the SFT (Set File Type) list of filetypes used by the Set FType command.

Edit Adj List... Brings up the File Type Editor allowing the user to edit the AFT (Adjust File Type) list of filetypes used by the Adj All FType command and the built in filetyping function.

The dialog which is displayed appears as follows:



The information in the area at the top of the dialog displays the details of the item selected in the list box at the bottom of the dialog. As the user selects different items in the list at the bottom, the info displayed in the boxes at the top changes to match the selected item. Not all entries will necessarily have all of the info listed. The information shown is as follows:

App	Application name belonging to this file type and creator (with English language type appended to make types easier to find in the list).
File Kind	English language name of file type
Type	System filetype for this type of file
Creator	System creator for this type of file
Extension	Extension commonly used for this type of file
Post App	Post Processing App used for this type of file (unused)
Post Crea	Post Processing App's Creator type used for this type of file (unused)
Mime Type	Mime Type used for this type of file

Dialog Commands and Options

Sort By — The list can be sorted by any of the parameters shown simply by clicking on the appropriate button.

App Name	Sorts list by Application Name
Type	Sorts list by file Type
Creator	Sorts list by file Creator
Exten	Sorts list by Extension

Iconfig — Internet Config related Controls at the top

Load	Loads the list from the Internet Config list (very large at 2-300 entries - should not be done with AFT list or the user will end up with a lot of work to do to undo the change). Somebody is bound to be curious enough to do this so here is a tip for an easy way out of that dilemma - just use the Load Default Lists command on the Edit Menu to restore the original lists. All user entered additions will be lost however.
Look Ext	Performs a lookup in Internet Config for the filetype corresponding to a specific extension which the user has entered in the Extension text box. The results are loaded into the item details boxes at top.
Look Crea	Performs a lookup in Internet Config for the filetype corresponding to a specific file creator which the user has entered in the Creator text box. The results are loaded into the item details boxes at top.

Buttons — The buttons at the right control both the list box and the selected item details above.

Clear	Clears the item details
Insert	Inserts the item details at top into the list in front of the item selected in the list at bottom.
Delete	Deletes the selected item in the list at bottom.
Change	Changes the selected item in the list to the values in the item details at top. The user can select an item, change the details at top, then click Change to enter those changes into the list.
Append	Appends the item details at top into the list at bottom after the last item in the list.
Example	Allows user to browse for file of desired type, then enters that information into the item details at the top of the dialog. To add your own filetypes, click example, find your file, click OK (in file browser dialog that example brings up), enter an extension, click append, click OK to accept and dismiss.
Cancel	Cancels changes made to the list and dismisses dialog.
OK	Accepts changes made to the list and dismisses dialog.

Rearranging List — To rearrange items in the list, click on an item to select it (release mouse), then while holding option key down, click and drag the item to the location desired. The list will autoscroll with variable speed depending on how far out of list box the mouse is moved. A gray line across the list will follow your movements to show where the item will be placed when you release the mouse button.

For the AFT list, the order of appearance of items in the list is important and unnecessary sorting is discouraged. The list is searched from top to bottom to determine filetypes from extensions so it helps to have frequently used filetypes near the top of the list, but this is not always the case as you will see with the example below. In the AFT list, each and every item must have an extension. If a search is performed and the extension is not found then the last filetype in the list will be used to type the file. Therefore it pays to have some kind of generic file type (like SimpleText .txt files) located as the last item on the list. Furthermore, since some files may have more than one extension, it is possible to have a situation where the wrong extension is detected depending on the order in which the user places the items in the list. For example we may have a file named as follows:

DonaldQuacks.mpg.sit.1

The filetype matcher will always detect the first occurrence of the extension string in the filename text if there happen to be two of them (as in DuckPic.sit.2.sit) although this is of no consequence. the thing to remember is that this extension matching operation is performed for each extension in the list and in the order in which each filetype occurs in the list. When a match is found, this process ends and the filetype corresponding to the matched entry is used for that file. The only exception to this rule is in the case of the Adj All FType command if the “Match At End” option is checked. In this case, the search for a match will continue until a match is found AND that match occurs at the end of the filename. The “Match At End” option does not apply in the case of the built in filetyper simply because legitimate filenames can exist with text following the extension as in the case of a duplicate file (e.g. DuckPic.jpg.01). The process of finding a match proceeds as follows using our example file from above (DonaldQuacks.mpg.sit.1) and the list in the illustration. Note that for each extension, an attempt is made to match that extension using both a “.” and a “_” as the leading character of the extension - in some cases you will see extensions separated by the “_” character and this ensures that we do not miss those odd names.

Does the filename contain “.aif”	If not then continue
Does the filename contain “_aif”	If not then continue
Does the filename contain “.au”	If not then continue
Does the filename contain “_au”	If not then continue
...	
Does the filename contain “.mpg”	If not then continue
Does the filename contain “_mpg”	If not then continue
Does the filename contain “.sit”	If not then continue
Does the filename contain “_sit”	If not then continue

In our case of a file named DonaldQuacks.mpg.sit.1, the search will end when it reaches the test shown of Does the filename contain ".mpg" and the file will be typed as an mpg file. This is incorrect by the usual order of file naming where each additional layer of processing adds another extension at the end of the previous name. This file SHOULD be a .sit file but with the filetypes ordered alphabetically as shown in the list, this cannot happen.

This is where the user ordering of entries in the AFT list becomes important and will determine the results which are obtained. The list as it exists in the illustration has obviously been sorted at sometime by extension, so the user will have to do this on his own if I do not get around to doing so with the default lists first. Fortunately, common sense dictates that there is always a certain order in which extensions will occur where mixed. For example it would not be too uncommon for mpgs or jpgs to be stuffed into a .sit archive, but you will never, ever find a .sit archive which is subsequently jpeg encoded. The easiest way to properly organize the AFT list is to simply use it while noting any discrepancies with files that have multiple extensions as in the example above. Whenever any such discrepancies occur (as noted by the wrong filetype being assigned), simply edit the AFT list and move the desired filetype entry to a position in the list above the erroneously assigned filetype. in our example case of "DonaldQuacks.mpg.sit.1" we would simply move the .sit entry to a position above the .mpg entry and all will be well.

Pattern Matching Syntax

This information has now been moved to a separate included document named “Pattern Matching”. It has been separated in order to also serve as separate documentation for other programs which use the same pattern matching libraries and also as a stand alone tutorial for novice users of regular expressions. It is better to maintain one document for all than individual separate copies. See that document for further information.

Miscellaneous Notes

Note about Joining Files:

The Join command is really intended to be used for joining segmented multimedia or any other large files (.mpg, .avi, .mov, .wav, .mp3 etc), but it can also be used to join the data forks of any kind of raw binary files by turning off both the “Only .nnn Files” and “Start on .001” options. A newbie user who knew nothing at all about multimedia files once wrote to me asking why MNT would not join the mpeg files he got at some web site (but he refused to give me the address so that I could test it for myself). I finally figured out that he was trying to use MNT to join two complete mpeg movies, for example:

He wanted to join:

MickeyMouseBuysCheese.mpg
DonaldDuckGoesFishing.mpg

To create:

MickeyAndDonaldEatDinner.mpg

(See chapter on Usage tips for a quick and dirty way to do this using Apple’s QuickTime Pro Player).

MNT’s Join command and other simple raw data concatenation commands of any program are not designed to handle such a function. What you need in order to perform such an operation is a movie / video editing program such as Apple’s iMovie, Adobe Premiere, Apple’s Final Cut Pro, QuickTime Pro Player or my favorite, a small shareware program called QuickEditor (see http://wild.ch/quickeditor/). Different movies can and often do have differing screen widths, heights, color depths, compression methods etc etc and simply joining the raw data does not at all ensure that the movies get joined properly. On the darker side it is even more likely to result in a jumble of raw data that is totally meaningless. In order to do this properly, you need a movie editor which reads the actual movie data and combines it frame by frame in a proper manner.

It is conceivable, however, that you may have two separate movie files such as those listed above which DO have identical basic parameters. The mpeg format is remarkably resilient as to how extensively you can totally clobber its data while it remains able to sync up on the new data and to present the movie properly. In such cases of identically formatted movies, you might be able to join the two by turning off both the “Only .nnn Files” and “Start on .001” options before performing the join and end up with something that might actually play. I would not try this on anything other than .mpg or maybe .avi movies because other formats (like QuickTime .mov movies) are much more strict and are unlikely to come through the process unscathed. The best choice, however, is to not try this kind of slipshod operation at all but instead to use the proper tools for the job. After all, your mom DID teach you that you cannot eat soup with a fork, didn’t she?

MIME Decode/Encode

All of the MIME related routines were completely rewritten for version 1.2 of MNT to become simpler and more generalized for sake of supporting as many MIME types as possible. The basic MIME engine is now built around only 5 functions with a recursive design that makes support for nested MIME structures totally inherent. It should be possible to decode nested MIME messages to any depth constrained only by the memory that is available to the decoder. This handful of basic functions are very modular making it possible to easily add new MIME types as needed in the future. MIME is a highly extensible standard and dealing with all of the possibilities can sometimes become quite complex. So much so in fact that there are actually some programmers who spend their lives specializing in nothing else but MIME. For this reason it is impractical for MNT to provide support for every conceivable MIME feature without devoting an equivalent amount of time to the effort (perhaps half of the time given to this revision has already gone into work on the MIME related stuff as it is). Instead, I have tried to thoroughly cover all of the basic types likely to be encountered in working with binary files (plus a few that nobody has ever considered or heard of) while leaving a few others with minimal or no support. In particular, there is no special support for the “message” content type with its associated subtypes and parameter tags. The “message” content type is detected but the data is treated only as raw binary data. This support may be added in future “visits” to the MIME functions. In the unlikely event that anybody is really looking for some program to process such “message” content types, you are likely to have better luck with a program designed to do nothing but MIME processing such as Mpack (free from CMU).

Comments Regarding yEnc

See additional important comments in chapter on “Future Additions”.

Anyone who has spent any time at all in any of usenet’s binary groups is only too familiar with the recently developed yEnc format for binaries encoding. I have some very strong opinions about yEnc, primarily regarding the strong arm tactics used to impose its use on the many “little guys” out there who disapproved and regarding the foul attitude of the emissary sent by the yEnc team to introduce it to the world while insulting and belittling all of those who complained. I will not include my full opinions here because the yEnc people would certainly list me on their blacklist page if I did. However, anyone who needs a sympathetic shoulder to cry on is welcomed to email me. Generally speaking, my technical opinion of yEnc could not be better expressed than has already been done by Jeremy Nixon on his web page listed below.

Why yEnc is bad for Usenet - Jeremy Nixon, the real originator of the yEnc concept speaks out

See web page at http://www.exit109.com/~jeremy/news/yenc.html

Technically speaking, the underlying concept is an excellent idea. The yEnc encoding scheme itself is extremely simple (indeed much simpler than uuencoding or base64), very fast, efficient and easy to implement. However, the web of the yEnc encapsulation mechanism which has been spun around the basic yEnc encoding method leaves something to be desired. Specifically, and from the standpoint of MNT, there is at least one ambiguity built in to yEnc, as defined, which forces any programmer implementing yEnc to make one or more assumptions which will affect the end behavior in processing yEnc data. There is a nice feature in yEnc (already existing for message types in Mime) which allows it to decode multisegment binaries in which the encoded segments arrive totally separately and are processed at different times. For lack of a better name I will simply call these multipart posts. The problem is that with such encoded segments arriving at varying times, there is no way provided to associate a given segment with its intended target file other than through the filename itself. The programmer can either take the approach that one highly touted yEnc program does and to avoid implementing this feature altogether or he can try to implement it with the possibility that something else is going to break on some other kind of yEnc encoding at sometime in the future when the decoder receives data which does not fit a stricter interpretation.

Lets take the example of a pair of yEnc binaries posted on the same day at nearly the same time. This is far more common than it may at first seem to be. Sometimes lazy posters give their attachments a series of incrementing numbers as filenames rather than using the attachment’s true filename. This is especially true of those spammers who post images containing advertisements for their “pay me” websites plastered all over the original image. Most of such posts are pure trash, but this is not always the case. If posted as multipart yEnc, this might give us the following series of segments in the newsstream:

```
11.jpg.01       First Post
11.jpg.02
11.jpg.03

11.jpg.01       Second post by another user at same time with same name
11.jpg.02
```

The user might download and process all 5 segments at the same time, so for all practical purposes the segments must be capable of being processed together in any order. The decoder is going to be totally confused because it has no means of separating the segments of the two posts. It may or may not be able to detect that something is wrong based on the size, begin and end tags and will certainly be able to detect any bad value for the final crc32 tag, but there is no definite mechanism provided by which it can reassemble such posts correctly unless they do arrive (and get processed) in a specific order. The solution is really quite simple (and I intend to suggest this to the yEnc people). If each such multipart post were required to provide a unique message ID as part of either the ybegin or ypart line which would be consistent across

a given binary's segments, then this ID could be used along with the other tags provided to positively identify a given encoded segment with its correct target file and byte location within that file. The format of this suggested message ID is not important and should probably even be left unspecified and at the option of the implementor (although it should probably exclude space characters so as not to interfere with detection of the boundaries of the other tags). What is important is that something needs to exist to identify a given post across the range of its segments.

In lieu of a message ID tag, two possibilities currently exist that I know of which could be used to resolve this problem, each with its own disadvantage. The ideal identifier for distinguishing such like posts would be the document crc32 value for the entire file. It is unique (with a 1 in 4 billion chance that separate files will produce the same value), however it is found only in the final segment of such multipart posts. Furthermore it is located in the yEnd line at the end of the segment, forcing the program to read the entire segment to get the crc value in order to determine the appropriate target file and then to have to rewind the file and repeat the read to decode the data. Furthermore, while the pcrc tag is required, the crc32 tag for the entire file is only optional, so there is no guarantee that it would be found at all. After some experimentation with this approach I got mixed results and finally had to disable its use.

Another possibility would be to use all or part of the Subject line as an identifier. The disadvantage here is that although it is likely, there is no guarantee that the Subject line of two posts are different from each other and consistent across all of its own segments. Furthermore, there is no guarantee that a Subject line actually exists for any given segment; there are no subject lines where files do not originate from a communications related system. The yEnc teams own "novus" test files are an example of this.

The last possibility would be to use the filename extracted from the yBegin line as an identifier. This will always be present, however it is the original source of the ambiguity described above and will fail in the presence of posts of two unique files of the same name.

To improve the efficiency of processing single part posts, it would be possible to turn on the association of segments with their appropriate target files only for multipart posts and to rely solely on the filename as usual for single part posts. The appropriate thing to control this option would be the presence of the ypart line in any given encoded segment. However, it is becoming a common practice for newsreaders to always include the ypart line (probably for sake of simplicity... or perhaps laziness) regardless of whether the segment is single or multi part. This would force the mode of operation to the slower multipart mode all of the time, causing such a multipart detection method to fail. I can think of other ways to do this, but the choice of such slow or fast methods is relatively unimportant compared to associating the correct segments with the correct target file for multipart posts.

Personally, I think that a message ID tag is the real solution to this problem. For those developers who choose not to implement such multipart features, the message ID could simply be ignored. Hopefully, the yEnc people will choose to adopt such a feature.

yEnc encapsulation as presently defined

```
=ybegin part=1 line=128 size=25431 name=novus.jpg
=ypart begin=1 end=5120
...
=yend size=5120 part=1 pcrc32=21F53E5E crc32=C62552CC
```

yEnc encapsulation as suggested
The CRC value is used as ID here but this format should be at implementor's option

```
=ybegin part=1 line=128 size=25431 ID=C62552CC name=novus.jpg
=ypart begin=1 end=5120
...
=yend size=5120 part=1 pcrc32=21F53E5E crc32=C62552CC
```

Beginning with MNT version 1.2.1, the yEnc decoder has been rewritten to always operate in multipart mode rather than trying to make assumptions about whether a given post is complete or not. This is slightly slower, but almost totally foolproof. Provisions have been made to accomodate a Message ID tag if it should ever become a reality. In its absence, a Message ID is derived from either the subject line or the filename itself in that order until a valid ID is established. This is used as a temporary filename for accumulation of the output data and miscellaneous decode accounting info until it is determined that the temporary file is complete at which time it is renamed and typed appropriately. This works like a charm with one exception. In those cases where decode errors occur and a file remains incomplete, there is no way to know when the user has given up on trying to decode a given attachment. Due to the indefinite nature of multipart file arrival times, the temp files are ALWAYS considered to be valid towards the possible future arrival of a final segment and can never be legitimately deleted until the user wishes to do so. For this reason, there can sometimes be an accumulation of incomplete temp files in the decode folders when decode errors occur and it is up to the user to manually delete these temp files when he decides that they are of no further use. These temp files are easy to spot because 1) they all begin with a bullet '•' character, 2) they all end in ".y" and 3) they are all given the MNT partial file icon shown below (you must rebuild your desktop after installing MNT 1.2.1 to see these icons - restart while holding down Command and Option keys until you get the rebuild desktop dialog then click OK for the volume where you have MNT installed). With version 1.2.1 a feature has been added to optionally delete all such temp files in the decode folder at the end of processing of a given dropped item to automatically deal with this nuisance in cases where multipart posts are not being processed. See description of the Decode command for further info.

MNT Temp File Icon

Note that there is one PC news program (some version of Agent if I remember right) whose posts will always result in such leftover partial files even though the post has been decoded correctly. This is due to a flaw in the generated posts whereby each yEnc part is duplicated (this also cuts decode speed in half). For example if we have a 4 part yEnc post of a file named dog.jpg, it would generate parts which we will name as follows:

dog.jpg.1
dog.jpg.1
dog.jpg.2
dog.jpg.2
dog.jpg.3
dog.jpg.3
dog.jpg.4
dog.jpg.4

MNT would decode this OK, but when it had finished decoding the first copy of dog.jpg.4, it would look at the temp file, see that it was complete and rename it as dog.jpg as normal. However when it saw the second copy of part dog.jpg.4, the process would begin again as it should with the creation of a new temp file for dog.jpg. The remaining parts would never arrive and the temp file for dog.jpg would remain in the decode folder until the user deletes it. The bottom line is that if you begin to find such temp files in your decode folder and you know that there are no further parts of multipart posts to be processed, simply delete them because they are of no further use.

MacNetTools - How it came to be

MacNetTools began its existence as a simple application written for a friend who needed a way to find and catalog files which had no comments attached to them. Soon after that I happened to be muddling around in one of Usenet's multimedia newsgroups where I found a great deal of confusion among Mac users as to how they could join segmented files like the Wintel users could do. As it turned out, it could be done using minor features of several Mac utilities, but the capability was not exactly advertised to the public and was not the easiest thing in the world to do. I asked myself why should it not be so simple as taking a folder full of such file segments and simply dropping it on a Mac utility which would rejoin them into the original file. I set out to create such an application which only took a couple of weeks to create, but was not yet willing to release it to so many people without some rudimentary form of documentation. In the meantime I was so pleased with the results that I decided that such a utility should also have the capability of doing UU and Mime encodes and decodes, but I considered those to be future features.

It was at this time that I needed to do just such a UU encode to test something which has long since been forgotten. I started up the readily available Mac utility which does such encodes and was immediately annoyed at the way it promptly executed a time delay while reminding me that I had not registered it yet. For Pete's sake, gimme a break!!! Asking internet users to pay for such commonly used and simple functions is like having the IRS levy taxes on the lunch money that poor school kids use to buy lunch with!!! Are these developers so greedy that they readily stoop to the level of a common schoolyard bully? My opinion is that simple software functions like that which everybody needs to perform and which only take someone a few weeks to create should be offered to the public free of charge and not used as a tool to nickel and dime the public for profits like the phone company is so notorious for doing. Even Netscape is free and take a look at all that it does. It was at that time that I told myself that all of my other more important projects would have to wait because I was going to make the time to create such a utility program which would be a better mousetrap and to offer it to everyone for free. At that time all of these features were spread out among at least three of my projects so I decided to combine them into one program which after a lot of thought was given the totally unoriginal name of MacNetTools. If anyone has a better name for it, I am open to suggestions.

Usage Tips

Move Duplicates Usage Tip

Lets say that you have two folders named Cat and Dog in which you know (or suspect) that there are a number of duplicate files. You wish to give precedence to retaining the files in folder Dog over the files in folder Cat for whatever reason (one good reason might be that you prefer the names that you have given to the files inside of folder Dog if they happen to be different). To do this simply place it inside of the preferred folder prior to running Move Duplicates. That is, we would place the folder Cat inside of Dog before running Move Duplicates on Dog. This will force the folder Cat (and its contents) lower in the file scanning order so that all of the table entries for the files inside of Dog will occur before any of the entries for the folder Cat. Since the first occurrence of each duplicated file is always retained, the files in Dog get first chance of being kept over those in Cat. There are many variations of the technique behind this particular tip. Just use your imagination while remembering the following. The file scanner is configurable so that it may evaluate files in different orders but as used in MNT it always makes two passes on each folder and subfolder. On the first pass it processes all files within the main folder and then on the second pass it processes each subfolder in the same manner (main files first followed by files in subfolders of the subfolder). Within all groups, the files and sub folders get evaluated in alphabetical order and the first file (of duplicate files) seen by Move Duplicates is always the one that is retained.

Joining Movies - How To Combine Apples With Oranges

A couple of days ago I found a tip in one usenet group that is the answer to the prayers of all newbies who think that any file concatenation tool can join two or more complete movies (whether .mpg, .mov, .avi or whatever). As explained in the chapter on Miscellaneous Notes, this operation would normally require a movie editor. Apple's QuickTime Pro Player (with emphasis on the Pro version) is a basic form of such an editor and it seems that there is a relatively unknown feature of this program which allows two dissimilar movies to be appended onto one another, even if they are of different types, frame sizes etc. To do this follow these steps:

1) Drag the first movie onto QT Pro Player to open it
2) Position the time slider at the bottom of the window to the position where you wish the next movie to be inserted.
3) Drag the next movie onto the open QT Pro Player window. It will be inserted as desired.
4) Repeat steps 2 and 3 as many times as desired.
5) Select Export from the QT Pro Player File menu. Make your desired choices for the Export popup selection and via the Options button for that Export format, then click the Save button. the combined movie will then be saved as specified.

Known Bugs

Changes to be made before release if possible

1) Mime decoder still needs more work... but that will always be the case. :-)
2) Some parts of movable dialog windows do not get updated when another window is selected while dialog is active. This is now only true with the Move To folder boxes. These can be changed to controls but may not get done for ver 1.2.1.
3) Set FType Prefs dialog - Add, Delete, Chng buttons do not work properly. For now, use Edit FType command on Edit menu instead.
4) Filetype Editor dialog - when any filetype in the list is selected, the icon for that item disappears. This is a cosmetic issue only and will be fixed at some time in the future.

OS X Only Bugs

1) When running under OS X, file/folder drops onto MNT with MNT already running result in Finder window reactivate following dialog display, thus covering up the dialog. This cause may be that some MNT events may be getting higher priority than Finder events. In any case, the fix is not something that should be addressed at the last minute before release and will have to wait for later releases. Meanwhile, the workaround is to simply click on the Status window to bring MNT back to the front if this should happen.
2) System version is displayed improperly in About Box. This is new with OS X 10.2 and may be due to a change in the version word format.

Late Arising Bugs Fixed

1) All Fixed

Possible Bugs To Watch

1) File Tools dialog used to cause machine to intermittently do a hard crash (MacsBug could not recover). I subsequently did some work in this area and fixed one type casting discrepancy. Since then there have been no additional crashes. Will keep this one on the list for a while longer until even I am convinced that this bug is gone.
2) Folder Tools - drop with MNT already running does nothing - problem disappeared
3) Wildcard Match Check has at rare times caused machine to crash - MacsBug recovers OK. Problem seems to have disappeared.

Future Additions

1) The yEnc decoder is now vastly improved but there is still some “fat” remaining in the inner loops of the code which can be removed with a possible speed increase.
2) Further Appearance manager support, specifically the use of smaller fonts in all dialogs as opposed to the old Chicago/Charcoal 12 font. This is not so much an addition of choice as it is of necessity. MNT has so many options now that the available dialog “real estate” is becoming very crowded. Each additional feature brings with it the challenge of finding some way to fit the applicable controls into the dialogs and the obvious solution is to simply use smaller controls and associated fonts. In addition to being more practical, I think it would simply look better too.
3) Further expansion of mime support.
4) Change multitasking yields to a time based system rather than work based as at present. This should improve the consistency of cooperation with other tasks while maximizing the usage of the time that is available to MNT.
5) Notification that a task is complete if it completes while program is in background. Should have added this to this version but I never got around to it (although I DO have a round TUIT... really).
6) Additional Command Period abort support. Command Period is currently honored on each pass through the file scanner (it is built in) or once per file processed, but if an operation gets stuck inside the code that handles the processing of each individual item, the user is out of luck and must use Cmd Option Escape.
7) After working for 18 years in the field of radar measuring satellite position and trajectories, it is not surprising that one of my interests is geodetics and mapping. There is a lot of raw mapping data out there available for free from the USGS and of special interest is the high resolution DEM data which provides elevation and contour data for the entire US (and the world is coming in time). There is a neat (although crash prone) freeware Mac program which processes this data into a form that is usable by any 3D program but with a change in the USGS data format that came about with the higher resolution data, this program is no longer usable with the USGS data. When I find the time, I would like to add a set of tools to MNT to process such data into something that is usable on the Mac. These data files are typically rather large and the buffered data system that is built in to MNT would be ideal for handling such data with ease.

Version History

Ver 1.0 Released September 2000, Initial release

Ver 1.0a Released June 2001, Documentation Update - new web page and email

Ver 1.1 Released February 2002, First Carbon version prior to features update

Ver 1.2 Released October 2002, major features update and bug fixes

Ver 1.2.1 Released November 2002, intended to only be a maintenance and bug fix release but has turned out to be a major update in itself. The changes included the yEnc decoder fix, minor work on mime decoder to improve applesingle/appledouble processing, second binhex decoder added which kicks in if errs happen with the first and faster decoder (and work has started on a third binhex decoder), added a List Missing SIT segments, simplified the List Files options and fixed a filesize limit in the Macbinary decoder. Also the wildcard substitution code was rewritten to be simpler and much more powerful, two regular expression matchers were added as alternatives to the wildcard matcher and the match check dialog was redesigned. Then all those old old System 6 popup menus were changed to the standard System 7 popups that everyone else uses these days... they are slower but look better and work with the system more easily. Plus several other additional commands were added and a visual error indicator with error counter was added to the status window for user feedback, an option for Move Duplicates allowing choice of file to be kept not to mention all the documentation updates to go with it all.

Credits and References

Thanks, credits, acknowledgments and references to other useful associated or similar products and documents are as follows:

Apple Computer Inc. for the wonderful Macintosh itself

The following referenced items are trademarks of Apple Computer Inc.
Apple , Macintosh, Final Cut Pro, QuickTime, QuickTime Pro Player, iMovie, MacOS, Mac OS X

MetroWerks Inc. for the Codewarrior Pro Development System used to create MNT

Premiere is a trademark of Adobe Systems Inc.

UNIX is a trademark of Unix System Laboratories, Inc.

MS-DOS is a trademark of Microsoft Corp.

All other trademarks are held by their respective owners

Thanks also go to The Walt Disney Company for creating the many fine cartoon characters whose names served as sample filenames in this documentation.

MD5 Algorithm is from
RSA Data Security, Inc.
http://www.rsasecurity.com/

Carnegie Mellon University for Mpack 1.5
ftp://ftp.andrew.cmu.edu/pub/mpack/mpack-1.5-mac.hqx

DiskTracker
Mark N. Pirri / Portents, LLC
http://web.mit.edu/mnp/www/dt.html
http://www.disktracker.com/

List Files Fat / List Files 2.6
Alessandro Levi Montalcini
http://www.montalcini.com/

Checksum 1.3
Geoff Walsh
http://www.macinsearch.com/infomac/disk/checksum-13.html

A Better Finder Rename
Frank Reiff
http://www.publicspace.net/ABetterFinderRename/

Graphic Converter
Lemke Software
http://www.lemkesoft.de/

QuickEditor
http://wild.ch/quickeditor/

Regex Matchers

Regexp
Copyright (c) 1986 by University of Toronto.
Written by Dr. Henry Spencer @ U of Toronto Zoology
Both code and manual page were written at U of T.
They are intended to be compatible with the Bell V8 regexp(3), but are not derived from Bell code.

Regexs
Copyright 1989 by English Knowledge Systems, Inc. All Rights Reserved.
Original Filenames were SR.C, SR.H which were part of some unknown archive

Regular Expression Syntax (try the following links for more info)

http://py-howto.sourceforge.net/regex/regex.html
http://wks.uts.ohio-state.edu/unix_course/intro-73.html
http://sunsite.ualberta.ca/Documentation/Gnu/rx-1.5/html_node/regex_toc.html
http://cclib.nsu.ru/projects/gnudocs/win/gnudocs/regex/regex_toc.html
http://sunland.gsfc.nasa.gov/info/regex/Top.html
http://activedeveloper.dk/iishelp/jscript/htm/jsgrpregexpsyntax.htm
http://webdocs.caspur.it/ibm/web/vacpp-5.0/lpex/ref/rlrgxsyn.htm

CRC Documents

A Painless Guide to CRC Error Detection Algorithms V3.0
By Ross N. Williams, 19 August 1993
ftp://ftp.rocksoft.com/papers/crc_v3.txt
ftp://ftp.adelaide.edu.au/pub/rocksoft/crc_v3.txt
http://mail.symuli.com/CRC/crc_v3.html

Also See

CRC and how to Reverse it
Author: Anarchriz/DREAD
Release Date: 29 april 1999 (last modification 30 april 1999)
http://www.yates2k.net/anarchriz_crc.htm
http://www.yates2k.co.uk/anarchriz_crc.htm (sometimes not online)

Commonly used CCITT16 algorithm gives incorrect checkfile value
http://www.joegeluso.com/software/articles/ccitt.htm

Final Comments & Contact info

Since its original release two years ago, there has not been a huge response to MNT, but those who have responded and given me feedback have been overwhelmingly enthusiastic about what it was able to do for them. The original program was admittedly somewhat “klunky” but I have had many ideas for it and have just completed the work on version 1.2.1 which I believe you will find to be a major improvement over the original. My ideas just keep on rolling in, but for now this is where it stops.

I think I have managed to locate and fix most of the bugs in the program, but with use I’m certain that you, the users will in time find others. If you do find files which you cannot decode or otherwise process, I would appreciate it if you would send me copies of them so that I can use them to further debug the program for you. Please do NOT send me any files over 500K though. If it is larger than 500K, just tell me (if and) where on the internet it is located and I will get a copy myself.

I also welcome suggestions. About 10% of the commands in MNT are the direct result of user suggestions. If you have a feature or command that you would like to see added to MNT, then just tell me about it, but do give me enough detail in your description and examples of how such a feature might be used so that I will fully understand your request. I might not necessarily implement it exactly as requested but at least will try to give you something that can be used to accomplish the desired task. One of my goals behind the design of MNT is to try to make it as generalized as possible so that one command may serve many purposes. A good example is the rename command which resulted from a user request for a command to strip leading exclamation marks from filenames. Instead of doing specifically that, I made it possible through rename to change anything to anything else.

It is my intent that MacNetTools will remain a free program for all. Several users, however, have been so impressed that they have asked me how they could make a contribution for the time I have spent in developing MNT. I am not beyond accepting any voluntary contributions if there are enough of them to make it worthwhile (otherwise the tax implications make it more of a hassle than a benefit). Towards this end I will probably soon set up a kagi account so that those of you who wish to do so will have a way to make voluntary contributions, but I want to make sure that everyone understands that this is purely voluntary and that the rest of you should feel free to accept MNT with my compliments and best wishes. This will not happen soon enough for the kagi account info to be included in this documentation rewrite but will be posted to the MNT website late when it become available.

I currently have 4 major projects underway (including MNT which is the smallest of them). After doing the yEnc revision described above in Future Additions, it is possible that I may not return to any major work on MNT for a long time as my attention turns to my other projects. But who knows... MNT is so much fun to work on that I just might give it a few days worth of work from time to time in adding one feature or another. Time will tell.

Phil Rogers
nettools@nym.alias.net Primary email address
macnettools@hotmail.com Alternative email address - mail to this one is less often checked

http://ksinksw.tripod.com/ MacNetTools Web Site and Primary Distribution Source

Go here to get the latest info on update plans, bugs found, future features, current contact addresses and more. This web page will usually be more up-to-date than this document. As always spam can render any email address useless over time so this is always subject to change. The current address will always be listed in this document and perhaps soon on the web page itself in an image form which the spammer’s address harvesters cannot read.

- -