This is the manual for FiFi, the File Finder.

Structure of this document:

I - PURPOSE

FiFi (File Finder) is a retriever... It will search for, and retrieve, all files containing one or several strings (up to three). These strings may be combined, so that you can send FiFi off to search for files containing one string and/or a second string, but not a third one etc...

If you are not familiar with the pointer environment, please read the section on the pointer environment" NOW. As of here, some familiarity with the Pointer Environment is assumed. It is recommended that you have the Menu Extensions ((C) Jochen Merz) installed when using FiFi. These can be downloaded from Dilwyn Jones' or Marcel Kilgus' websites.

II - LOADING OR STARTING THE PROGRAM

FiFi can either be EXEC'd or loaded with (L)RESPR.

A - EXEC

The program should be EXEC'ed as follows

```
EXEC <device>_FiFi_exe
```

It can be put on a Hotkey.

B - RESPR or LRESPR

Loading FiFi this way should be done after loading the Hotkey System II. If FiFi is loaded that way, it installs itself as a "Thing". It can then be called with EXEP, HOT_WAKE etc directly.

If you have QPAC II installed in your system, FiFi can then even be called with BT_SLEEP, installing itself directly as a button. If it is set up with BT_SLEEP, then the 'ESC' item produces the same effect as the "Zzz" item, i.e. instead of disappearing altogether, FiFi goes to sleep behind its button.

III - THE INTERFACE

When the program is started, it comes up with its main screen, which contains all the menu options pertaining to its use. Generally speaking, the user typically will determine the strings to search for and the directory of the device on which to perform the search. Hitting the 'OK' item will get the search started. Once FiFi has gone through all files on the directory requested and its sub-directories, it will list the files matching the search criteria in a new sub-window and state how many files were found.

Describing the more exact use of the program comes down to describing the different menu items and options. From top to bottom and left to right, the following menu items are present.

A - Title row

This is the row containing the name of the program. From left to right, it contains the following items:

1 - Move Window (standard selkey).

2 - Change window size (standard selkey).
Note that this item only has an effect if FiFi has some filenames to show.

3 - Help (SOS) item (selkey : F1).
Please see the section on the integrated help system.

4 - "About" (selkey: A).
This tells you all about FiFi.

5 - "Macro" (selkey: M).
This lets you load or save 'Macros', i.e. command files. Please see below in the Command String section.

6 - Status (selkey : F5).
This call up a small status menu where you can change some options:

- Set the delay until FiFi show its help window please see the section on the integrated help system).

- Set the name of an output file into which the result of the search (i.e. the name(s) of all matching files) is to be placed. If you type in an invalid name, or the name of a file that cannot be opened or found, then the result will be shown on the screen as usual. If you want to display the found files on the screen again, simply delete the name of the output file in the F5-Status menu.

7 - ESCape from the program
Quit definitely.

8 - Sleep (standard selkey
Makes the program into a "button".

B - Second row

From left to right, this contains:

1- The start directory to search in (selkey: F2)
You must tell the program where it should start to look for the files to be searched. This is done via the F2 item which will bring up the directory name for editing. If you DO this item and have the Jochen Merz menu extensions installed (which is recommended), this will bring up the "Directory select window". If you HIT this item, you can just enter or edit a name.

The letter '*', if used as directory name, will be replaced by the data default directory. This is true here, or when configuring FiFi or wen using a command string or file.

Likewise, "$" is replaced by the program default.

Unless the Xsub option is selected, FiFi will search for matching files in this directory and in all of its sub-directories.

2 - "XSub" (selkey: X)
This item stands for eXclude sub-directories. Normally FiFi will search in the start directory and in all of its sub-directories. If this item is selected, FiFi will not search in sub-directories.

3 - "Word" (selkey: W)
If this menu item is selected, then a string will only be found if it is a "word". This means that the string must start and end with one of the following: a space, a newline, the enter keystroke ($0A), or any of the characters **,:;.!?**. The beginning and the end of file are also start and end markers for words.
Thus, if you search for 'hound' with this option set, then FiFi would not find, for example, 'greyhound'.

4 - "Space" (selkey: S)
If selected, this switches on a special treatment of spaces and Tabs during the search. In assembler, for example, one often formats the source code, for example:

```
label   trap    #$f
```

Normally, if one wanted to search for the "trap #$f" and find the above text, then one would have to insert the exact number of spaces between 'trap' and '#$f'.

FiFi can do better with the "Space" item. When this is selected, and a space is contained in a search string, then FiFi will find any file that, at the same place, contains not only one but several spaces (or TABs) - provided of course that the rest also matches!

Example: Suppose the search string is:

```
        trap #14
```

If a file then contains the string
```
        "trap   #14"
```
or even
```
        "trap         #14"
```
then this will be a valid match (if the Space item is selected), but "trap#14" will not be a valid match.

Thus, when a space is contained in the search string, FiFi continues checking the file until the first character that is not a space or a TAB.

Indeed, the same is also true for the TAB character, which is held to be just a combination of spaces: again, if the search string is

trap #14

then

"trap<tab>#14" will be a valid match

(of course, the <tab> is the tab character, i.e. chr$(9), and

"trap <tab> <tab> #14"

would also be a valid match.

If you select the Space item, it will work on all search strings. Never use several contiguous spaces in your search string if the Space item is selected:

"trap #14" would be a bad search string, it is guaranteed that no match will ever be found (you should be able to figure out why).

This space/TAB stuff does NOT work if the space is the first character in the search string!

When the Space item is selected, and FiFi calls QD upon a DO on a found file, FiFi only gives QD, as string to search, the string until the first space (or TAB).

5 - "Text" (selkey: T)

Normally, FiFi will search all files in the directory. If you are sure that your search string is in a text file (i.e. not an executable or rel file), or if you deliberately want to search text files only, you should select the "Text" item. FiFi will then disregard any file which is not a text file. This may speed up the search considerably (or not, depending on the number of non-text files in the directory). Non-text files are, of course, all executable program files, as well as some "_REL" files produced by some assemblers. Some Prowess files will also not be considered as text files.

6 - "Names" (selkey: N)

Sometimes, one isn't looking for a string contained in a file, but rather for a certain file with a certain name. If you select the "Names" item, FiFi will not look at the content of the files, but at their names. If they match the criterion used, they are reported.

C - Third row

From left to right, this contains:

1 - Extensions (selkey: E)

Here you now determine any number of extensions. The search will then be performed only in files having these extensions. The extension of a file is defined as being the part of the filename to the right of the last underscore "_".

If you DO the extensions item, it will call up the Menu Extensions' extensions menu (but this will only allow you to enter one extension).

If you HIT the extensions item, you ca enter/edit several extensions. You should enter the extensions in one single string, but separated by a character such as "_" or "," or ";" or ":" or a space, i.e. any kind of character which isn't part of a normal filename. You can indicate as many extensions as the window is wide - 56 characters.

If you set the NOT item next to the extensions item, then only those files will be searched that do NOT have any of the extensions shown.

2 - All (selkey: F4)

This simple selects/deselects all files found.

3 - Not (no selkey)
If this item is selected then only files NOT having the extensions set in the extensions item will be searched for/through.

Finally, there are the search strings and associated items (Case, Not, And, Or) which are treated in more detail below.

IV - THE INTEGRATED HELP SYSTEM

FiFi contains an integrated help system. There are two items associated with it : the help item and the status item.

A - The help item.

This (selkey: F1) switches the Help item on and off.

Select help (by 'hitting' or 'doing' this item. Then put the pointer over any other item you are not sure about. Wait for a few seconds **without moving the pointer**. A Help window for this item then pops up. To get rid of the Help window, hit ESC. If you want help again on the same item, you must move the pointer out of the item first, and then back in again!

The Help window has a curious new item, on the right. This is hit/done by 'X'. When you use this item, a 'DO' event is carried through the Help window (which disappears) to the underlying item (on which the help was displayed), so that it appears as if you had DONE that underlying item.

If you de-select the help item again, no more help windows will pop up.

B - The status item

The status item enables you to change the time until the Help window pops up over an item. At its lowest setting, the help window will pop up nearly instantaneously.

V - THE SEARCH STRINGS, THEIR OPTIONS AND COMBINATIONS

A - The search strings

Up to three search strings can be used (selkeys: 1 to 3). At first, these are empty. Hitting the empty space where they fit in (under the "xxx Searchstring" signs) will enable you to fill them in. Alternatively, hitting keys 1 to 3 will enable you to fill in strings 1 to 3. This is as it should be.

The following rules apply here: One cannot fill in string 3 if string 2 is empty, nor string 2 if string 1 is empty. It follows that one must fill in or edit string 1 first, then string 2, then string 3.

Should you do that, and then edit string 1 so that it becomes empty again, then all other strings become empty, too (this is a quick way to reset them all).

When setting the search strings, if you use the up and down arrow keys, then you go to the next/previous search string.

B - Searching for non-printable characters

It is possible to search for non-printable characters, such as chr$(0). To find such a non-printable character in a file, you must give as search string a '$' (without the apostrophes!) followed by the hex code of the character, to be entered in valid HEX and always with two figures (e.g. 01 or AB or 00).

Example: ABCD$01EFG
will search for the string ABCD followed by ASCII code 1 (e.g. like CHR$(1) in Basic) followed by EFG - the '$01' was simply replaced by ASCII code 1. The dollar sign ('$') denotes the start of a hex code.

If you want to search for the dollar sign itself, replace this by $$ (a double dollar sign).

For example: ABC$DEF
will find ABC followed by the hex code DE, followed by F.

but ABC$$DEF
will find the string ABC$DEF.

If you enter an invalid hex code after the dollar sign (e.g. ABC$-+K), FiFi will NOT show an error - but it probably won't find your string either!

Of course, you can search for multiple hex bytes, by preceding each with the dollar sign, such as ABCD$00$01$02EFG.

The exact character that is used to introduce the hex sequence ($ normally) may be changed when configuring FiFi. Use whatever suits you, but the rules set out above still apply: use the character once to start a hex number, twice to search for the character itself.

C - The CASE and NOT items

1 - The CASE items

For each string, you can determine whether you want to search for the string EXACTLY as it is shown (i.e. distinguishing between upper and lower case letters), or not. If you select CASE for a string, then the matching string must correspond EXACTLY to the search string. If the Case item is left available (not selected), FiFi will also find files containing the string in another case.

Example: The search string is "Here". File A contains the word "Here", file B contains the word "HERE".

If a search is made with Case selected, FiFi will only find file A (exact match). If Case is not selected, it will report file A (exact match) and file B (same word, but in another case).

2 - The NOT item

If this item is selected, FiFi considers that a matching file is a file which does NOT contain the search string.

CASE and NOT are individually switchable for each search string. A search with CASE selected will be (slightly) faster than a search with CASE not selected.

D - Combining the search for several strings with AND/OR

As you can see, strings 2 and 3 have two additional items: AND and OR. These are (hopefully) quite intuitive:

"String 1 AND string 2" means that both strings must be present in the file for it to be match. "String 1 OR string 2" means that any one of these strings must be present (or both). "String 1 AND string 2 NOT" means that string 1 must be present, and string 2 must not be present etc...

This is all very straightforward, the only problem might arise when combining three strings with different criteria. If you combine three stings, the rule is: "(string 1 CONDITION string 2) CONDITION string 3". This poses no particular problem if the conditions are the same (i.e. two ANDs or two ORs). If they are different:

"(string 1 AND string 2) OR string 3" means that the file, to be considered a match, must contain either string 3 or both of strings 1 and 2.

"(string 1 OR string 2) AND string 3" means that the file, to be considered a match, must contain string 3 and either string 1 or string 2 (or both).

You will have noticed that the OR is not an exclusive OR: "string 1 OR string 2" means that either (or both!) of these strings must be in the file, but not that only either one or the other must be present.

VI - SEARCHING FILES

A - Starting the search

The search is started by HITing or DOing the "OK" item. When searching for files, FiFi displays the name of the (sub)-directory it is currently working on, provided you have at least level 2 drivers, with "true" sub-directories.

The name is displayed in the item normally containing the name of the start directory . You will notice the recursive nature of FiFi: it goes down a sub-directory as soon as it finds one, and comes up one directory level when the sub(-sub -sub etc...) -directory is totally finished.

Of course, FiFi will continue to work even if its window is (totally or partially) covered. In that case, however, these names may not be displayed.

Once the search is finished, and if any matching files are found, FiFi opens the Filenames Window, showing the files corresponding to the search criteria. If so configured, FiFi can also put the names of the matching files into a results file (see the Configuration section).

B - Contract to a button during the search.

Normally, FiFi keeps its window open during a search. You may prefer that FiFi contract to a button during the search. This is configurable. Once the search is done, FiFi automatically reopens its windows. You can also configure the job priority during the search when used as a button, and whether FiFi should display the directories being searched in the title of the button (See the section on Configuration - Search).

C - Same or independent search job?

During a normal search, FiFi's items are unavailable and you have to wait for the end of the search before being able to operate on it. FiFi

can also be configured to use a separate search job. If you do configure it to use a separate search job, then FiFi will display, about every half second, the names of all files found so far. You can then already have a look at any of these files, by clicking on the filename, as usual.

The following rules apply:

To have FiFi search with another search job, you must:

1 - Configure FiFi to use this other search job.

2 - NOT have set FiFi up to print the names of the files found to a file.

3 - NOT have set FiFi up to contract to a button whilst working (see below).

If any of the above conditions are not met, then FiFi will just continue to work as it did before - it will NOT warn you of this. If FiFi doesn't do the expected, make sure that the above conditions are fulfilled.

If you set up FiFi to use another job to do the searching, then, as soon as you hit OK, most of the items become unavailable - you cannot change the search strings or search directory whilst the search is already going on!

If you must, you can change the status of the CASE, AND, NOT, OR and SPACE items - these changes will be taken into account for the next file to be searched.

You can also change the status of the TEXT and NAMES items - but this will NOT be taken into account. Only the status of these items when the search was started is taken into account.

IF you hit/do the ESC item, which remains available, the search will be aborted.

As soon as FiFi's search job has found the first file meeting the search criteria, FiFi will open up its Filenames Window (see below) and show the first filename. And then the second, third etc... as and when they are found.

You can now click on any of these, and see the file/call up QD as usual. HOWEVER, if you do that, or if FiFi's window is covered, AND IF A NEW FILE IS FOUND, THE SEARCH WILL STOP until you go back to FiFi.

VII - THE FILENAMES WINDOW

When FiFi has finished its search, it normally brings up another window, the "Filenames" window, in which the names of all matching files are shown. FiFi will not open the Filenames window if:

- no files were found, or
- the results of the search were to be sent into a results file.

If the pointer is in another area of FiFi, the TAB keystroke will bring it into the Filenames window. Hitting TAB again brings the pointer into the scroll bar (and back again when you hit TAB again).

Once the pointer is in the Filenames window, you may HIT or DO a filename.

HITing a filename will send this filename into the Hotkey stuffer buffer (from which you can normally retrieve it with ALT+SPACE), provided you have opted for this when configuring FiFi (see below, the section on Configuration).

What happens when you DO a filename in the Filenames window depends on how FiFi is configured : FiFi can call up QD, or try to open the file via FileInfo or FileInfo II, or show the file in a Preview window.

A - The Preview window

FiFi opens up the Preview window and displays the part of the file where it found the matching string. Of course, if you gave FiFi several search strings, it is possible that they are located in different parts of the file, which cannot be shown together on the screen. Consequently, FiFi will only show the part of the file where the last string searched for is found.

If you have given strings which must NOT be included in the file, FiFi of course cannot show you where they are in the file (as they are NOT there), so it just displays the first few lines of the file.

If you have more than one filename selected, then you can see the next and previous files by hitting the "<<" and ">>" items.

B - QD

You can use FiFi so that DOing a found file calls up a new instance of QD and loads the file into it. This only works when QD is installed as a thing. If you have several files selected, then a DO on one of them will open a new instance of QD for each of them.

If you have a later version of QD (>6.07), then QD will automatically go to the first occurrence of the first search string which is not excluded by NOT.

C - FileInfo

When so configured, FiFi will try to execute a file you DID in the Filenames window via FileInfo. This of course presumes that you have the FileInfo thing loaded. If you have several files selected, then a DO on one of them will try to execute every one of them via FileInfo.

D - The ALL item (selection key: F4)

The main window has the ALL item (selection key: F4) which will select or deselect all of the names found.

VIII - COMMAND STRING

It is possible to give FiFi a command string. The command string can be composed of several commands, sometimes followed by some parameters. Each command must be preceded by an inverted backslash ('\').

A - Description of each command

-> \A: the "nAmes" item will be selected, a search is to be made in the names of files only.

-> \B: The "word" item will be selected.

-> \C1: set the CASE item for the first string
\C2: set the CASE item for the second string
\C3: set the CASE item for the third string
The default is no case, so by including these options, CASE dependent search can be required for any string.

-> \D string: determines the device/directory on which FiFi is to start working.

You should take care that the directory is a valid name, else FiFi will not find it. Do not leave a space between the \D and the name.

Example: ex FiFi;"\dwin1_my_directory_"

-> \E selects the space item.

-> \F string: (stands for result File) where string contains the name of a file into which FiFi will place the result of the search. This means that FiFi will not display the found files in the window, but will dump the names into the given file.

Due to the combination of the \R, \F and \W commands there is a possibility to start FiFi completely from the command line, and have it work entirely in the background.

-> \G: This must be followed by the extension name(s).Please note that, when setting the extensions string, do NOT use the '\' character to distinguish between the extensions...

-> \H: The "not" item for the extension will be selected.

-> \N1: set the NOT item for the first string
\N2: set the NOT item for the second string
\N3: set the NOT item for the third string
The default is that NOT is not selected, by using these option, you can set the NOT item for any of the strings.

-> \O2: the second string will be ORed (and not ANDed, which is
the default).
\O3: same for the third string.

-> \R (stands for Run): will cause FiFi to Run automatically as soon as it is executed, without waiting for the "OK" item to be pressed.

-> \S1<string>: This is the first search string.
\S2<string>: This is the second search string.
\S3<string>: This is the third search string.

ATTENTION: care should be taken NOT to leave a space between the command (\S1) and the string, else FiFi will think that the space is part of the search string. DO NOT include the square brackets < and >, which are used to make things readable only.

ex: EX FiFi;"\s1my_string" and NOT: EX FiFi;"\s1 my_string"

-> \T: the "Text" item will be selected, a search is to be made in text files only.

-> \U: The "Xsub" item will be selected.

-> \W (stands for no Window): means that FiFi will not display any window at all, not even to show the result of the search. This doesn't make much sense unless combined with the \R option and the following \F option...

-> \X string (stands for eXternal command file): you can give a name of a file containing a command string. FiFi will then read the content of this file, and use it as command string.

The purpose of this option is to let you compose an often-used command string and place it in a file. This avoids having to re-type the command string every so often.

The commands in the eXternal command file are the same as those to be used here. FiFi will only read one string in this file, which must be terminated by LF (i.e. a string produced by most, if not all, QL text editors). The string should not be longer than 800 characters (actually it is impossible to have a valid command string that long...), it will only be read in up to the first LF.

-> \Z: If this option is set, FiFi sets up as a button, waiting to be woken. Moreover, ESC then acts like "Zzz", i.e. FiFi does not disappear, but goes to sleep behind its button.

B - General rules for the command string

1 - Spaces

You should avoid putting spaces into the command string (e.g. to separate the items). Normally, the parser is intelligent enough not to take spaces into account, but consider the following example:

ex FiFi;"\s1coucou \dwin1_my_dir_"

The parser just cannot know whether you want to search for "coucou" or for "coucou " (with a space at the end), so it takes the latter option...

2 - The \ separator

2) The "\" will always be considered as the start of a new command. This means it cannot be included in a search string (there is no escape character for this).

3 - Case

The case of the commands themselves is not important, so "\N" is equal to "\n". Of course, the case of the string to be searched may be important.

4 - Parsing the command string

The command string is parsed sequentially from left to right. Consequently, if conflicting options are used, the last used option will prevail.

An illustration of this are the "\Z" and "\R" commands. These are potentially conflicting, as FiFi does not run when set up as a button. So, whatever was latest will be obeyed.

This also means that you can include some commands in the normal command string, and include as last item an \X option: FiFi will perform all commands in the string, and then (and in addition !) use the eXternal command file. You can even include an \x option in an eXternal command file, thus chaining several command files...

However, please note: any option to the right of an \X option is lost to FiFi, as, when it loads in a new command string from a file in compliance with an \X directive, the old command string is forgotten.

5 - Respect the string order
If you set string 3 without setting strings 1 and 2, this is an invalid option, the string will not be set. Likewise for string 2 if string 1 has not been set.

6 - No unset
There is no provision to "unset" the names or text items when they have been configured to be set. If you use FiFi from the command line, it might be a good idea first to configure it to unset everything, then setting the individual items per command string. The directory determined in the command string supersedes that which was configured.

C - How to get the command string into FiFi

This can be done in several ways:

1 - Direct passing of command string
The (TK II) EX and EW commands allow you to pass a string to a program, which can be used as a command string (this is what was used in the examples above). The command string is separated from the program name by a semicolon (;) and must be between quotes.

(Note: if you have TK II, then EX and EXEC are strictly equivalent, as are EW and EXEC_W. If you haven't got TK II, you should get it right now...).

Example:
EX win1_progs_FiFi;"\s1firststring\s2second\dwin1_asm_FiFi_\r"

2 - Through an external command file
FiFi supports not only the '\x' option described above, where the name of an external command file is contained in a command string, but also the other option foreseen by QDOS, i.e. having an input file channel ID put onto its stack. This is again achieved by the EX or EW commands, but the name of the file is separated by a comma (,) and not a semicolon.

Example: EX win1_progs_FiFi,'win1_command_file'

Do not use the comma and semicolon options together: if FiFi detects that an input file was opened for it, it will not look foe a command file...

Note that the parameter thus passed to FiFi is is NOT a command string, but simply the name of a command file, i.e. a file containing the command string.

FiFi may even be configured to use a default external command file, to be executed (nearly) every time FiFi is started, even if FiFi is set up as a thing.

You could thus set up several FiFis under different names, each with its own default external command file, to do repeated searches...

The default external command file will NOT be obeyed if you execute FiFi with a command string or an open channel to a new command file, as, then, the command string, or the new command file, will be performed.

3 - Loading it as a macro
The Macro menu item lets you load or save 'Macros'. A macro is nothing more than an external command file.

You can thus save the current state of FiFi as a macro, after you have selected and determined everything (e.g the search strings and directory, or the AND/NOT items) as you wish. Later, you can recreate this state exactly by re-loading the macro thus saved.

If a macro contains the \R option, it will be executed directly upon loading! Please note, however, that this option cannot be saved, but must be added to the command file later, with a text editor.

IX - CONFIGURATION

To configure FiFi, use the standard MenuConfig program which you can get from Marcel Kilgus' or Dilwyn Jones' site.

FiFi has four different configuration sections:

Colours - here you can set the different colourways for FiFi's windows.
Presets - here you can preset different menu items to be selected.
General - some general configuration options.
Search - some options about how FiFi proceeds with the search.

Please note the "cutoff when saving" possibility that MenuConfig offers for FiFi:

A - CUTOFF when saving after configuration

FiFi implements the possibility to cut off the config block (and certain texts used only in the configuration) if used with MenuConfig.
Attention, the help texts will then also not be saved and thus they will not be included in the saved version either.

This means that the configured file will be smaller than the original, but it:

- no longer has any help.
- cannot be configured again.

NEVER USE THIS OPTION (FROM WITHIN MENUCONFIG) ON YOUR MASTER COPY OF FiFi. ALWAYS USE IT ON A COPY OF THE PROGRAM.

B - Colours

You can configure the colours, i.e the way the windows look.

First of all, you can opt to use the colours as used by the Menu Extensions. This presumes that these extensions are loaded, of course! Please note that more recent versions of the Menu Extensions no longer have colours defaults. In that case, if FiFi is still configured to use colours defaults from the Menu Extensions, it will always use the old colourway 0, i.e. green/white.

If you opt to use the colour from the Menu Extensions, the rest of the colour presets become moot and are no loner available for editing.

If you do not opt to use the Menu Extension colours, you can define the colours for the main window and, potentially, the Filenames window, the Help window and the Button.

For the main window, you are given the choice of the following colours:

White/Green
Black/Red
White/Red

Black/Green
System Palette 1
System Palette 2
System Palette 3
System Palette 4

These are all standard Pointer Environment / SMSQ/E choices. The first four correspond to the old colourways 0 - 3, the others are the typical SMSQ/E system palettes. If you are unsure what they are, the best thing is to experiment, until you find the colour combination you like best.

If you set the main window to use any of the four system palettes, you can no longer set the colours for the Filenames window, the Help window and the Button, and any previous values set for them will be ignored - all windows will use the same configured system palette.

If you do not set the main window to use any of the four system palettes but configure it to use any of the old colourways, then you can set the colours for the Filenames window, the Help window and the Button. For each of these you will then only be given the choice of:

White/Green
Black/Red
White/Red
Black/Green

i.e. the old colourways.

C - Presets

In this section, you can preset some items to be already selected. For example, if you always search for words and thus select the "Word" item, then you can preset it here. the items that xan be set are cearly named suring the configuration.

D - General

Default directory, entensions and command file presets
You can determine what the default start directory name should be, and any extensions that should be preset. Likewise, the default name of a command file, loaded automatically at startup, can be entered here.

Character used as hex sign
Please see above, the section on Search strings.

What shall a DO on a found file call?(Filenames window action)
You can determine what happens when you do or hit a file in the Filenames window. You can choose whether a DO on a filenae in the Filename window should : call up QD, use FileInfo to execute the file, or just show it in the Preview window (see the section on tHe Filenames window).

Call QD or FileInfo from Preview Window?
If you elected to see a file with the Preview window, you get the chance there either to call up QD or to execute the file via FIleInfo.

Stuff name in Hotkey buffer when selected?
You can also determine whether a filename, when it is selected, will be inserted into the 'Hotkey' buffer. This can then be recalled by typing ALT + Space.

Make Sounds
You can configure FiFi to make sounds. There are three sounds:

1 - At the first file found (for the others, there is no sound).
2 - At the end of the search.
3 - At the end of the search is no file was found.

Sound 1 is a high sound, 2 a middle sound, 3 a low sound.

Determine how long FiFi waits before it shows the Help window
FiFi ca, wait a variable time interval the pointer has to hover over an item before the Help window is shown. Options range from 0 (veery short time) to 7. At its shortest setting, the help window will pop up nearly instantaneously.

Add current dirname to the sleep button ?
When FiFi in laid to rest behind a button (NOT when it is contracted to a button whilst searching, see below) it can display the start directory as well as the name FiFi in the button title. This makes it easier to keep several FiFis, working on different directories, as buttons, ready to pop up whenever you want.

Shall the window be auto-resized after a search ?
If you use FiFi for several searches, you may find more or less results on subsequent searches. With this configuration item, you can determine whether FiFi should resize its window to fit (as many as possible of) the search results in its window.

The options are :

1 - Always resize after a search.

2 - Only resize if the window needs to be made bigger to accomodate the search results.

3 - Always resize unless the window was manually resized. In this case, FiFi resizes the windows as it needs to, just like in case 1 above. However, once you resize the window manually, FiFi no longer resizes the window, just like in case 4 below (with the same exceptions).

4 - Never resize the window. Once a window size is set, FiFi doesn't change it anymore. The results of new searches will be displayed in the same window. If a new search yields less results, there will be empty space, if the new search yields more results, scroll arrows will be used. There are two exception to this :
a) after the very first search that actually yielded a result, the window is obviously adjusted to show the search results. The resulting window size is then the size FiFi will keep.
b) If a first search yielded only one search result, and a new search yields more, then the window will be expanded to show one result + scroll arrows.

Show file details of results ?
When showing the matching files in the Filenames window, FiFi can either just show the filename, or the filenames + details such as file size, file date and file version (not the file type, though). Here you choose which.

E - Search

Shall FiFi use a separate search job ?
As described in the section on 'Search' above, here you can confgure FiFi to use a separate search job (or not).

Priority of separate search job ?
Here you can set the priority of the separate job.

Shall FiFi contract to a button when working ?
It is possible to configure FiFi so that it contracts to a button when working. The button then shows the directory currently being worked on. Once finished, FiFi reopens its normal window.

Priority of job if set up as button when working ?
Here you can configure FiFi's priority (1 - 127) when FiFi contracts as a button whilst working: a priority of 1 will, of course, make FiFi slower, but will enable it to be called to the front by Button_pick, like any other button (this is already the case whenever FiFi is a 'normal' button).

VI - THE POINTER ENVIRONMENT

For those users new the the pointer environment, here is some information. The pointer environment (or extended environment as it is also called) is a new way of corresponding with the user of a program. User input is channeled through a "pointer" which appears on the screen. The pointer can be in the shape of an arrow, an image, a triangle, or whatever. In FiFi it is a small cross. The user thus "drives" the program. by bringing that pointer over some "items" (menu options) and then "clicking" on these items. This will then produce some action.

The pointer is moved with the cursor keys or, better still, with a mouse. When you bring it over an item, the item is highlighted, so that you know that this is the current item. These items can be scattered throughout the window. In FiFi the menu options "OK" and "ESC", for example, are items.

When the cursor is over an item, you can the click on this item. This will then produce an action. "Clicking" is pressing either one of the buttons of a mouse, or pressing the SPACE or ENTER key on the keyboard. By way of definition, if you press the ENTER key or the right mouse button, this is known as a "DO" (and the associated action is "DOing") whereas pressing the SPACE bar, or the left mouse button, is a "HIT" ("HITting").

Sometimes, there is a difference between HITting and DOing an item, sometimes there isn't. This depends on the program. itself. In FiFi for example, there is a difference between HITting and DOing the names of the files found.

Sometimes, items do not have a text in them, but a small "icon", or image, supposed to be a representation of what the item is supposed to do.

Items can have several statuses: "available", when it is possible to HIT or DO them, "unavailable" when, for any reason, the items cannot be HIT or DOne, and "selected" where the item shows that it is currently is use, or otherwise in a special state. The different statuses are shown in different colours. You will generally have no trouble in seeing the state any item is in. In FiFi for example, the "AND" items are selected when the program. comes up first. You can change the status of an item by HITting it.

Many items also have "selection keystrokes" Indeed, it sometimes can be a bit slow to bring the pointer over the item, and then HITting or DOing it, so some items can also be actioned directly by a keystroke. This is also useful for those who are allergic to the whole idea of pointers and mice.

Pressing the selection key for any item is the same as HITting the item. Generally, the keystroke is the first letter of the item (e.g.. 'o' for "OK") or any letter that is underlined in the item.

Some items are quite standard (or at least they should be!) throughout all programs written for the pointer environment.

These are:

* ESC (selection keystroke: the ESCape key). This generally quits an action, or the program. altogether.

* Change window position (selection keystroke: CTRL F4) is used to move the window around the screen. Its "icon" represents two windows, one on top of the other.

* Change window size (selection keystroke CTRL F3) enables you to change the size of the window. Its "icon" represents one small window within a larger one.

* Sleep (selection keystroke: CTRL F1) will put the program. to sleep in the form of a small button. This button can be "woken" by either Doing on it, or by the "Wake" keystroke. The "icon" for this is ... Zzz.

* Wake (selection keystroke: CTRL F2) will wake up a button, and generally ask a program. to refresh its menus. The icon for this item is a stroke of lightning.

You will also notice other effects of the pointer environment: where windows overlap, the content of the bottom window is preserved and appears in full when you switch into the job with CTRL C. This holds true even for programs that are not written especially for the pointer environment.

W. Lenerz

Last amended april 2018