ST_Modules

Michael Link

COLLABORATORS			
	TITLE : ST_Modules		
ACTION	*NAME*	*DATE*	*SIGNATURE*
WRITTEN BY	Michael Link	April 15, 2022	

REVISION HISTORY			
NUMBER	DATE	DESCRIPTION	NAME

Contents

Chapter 1

ST_Modules

1.1 Introduction

```
                Deutsch
               Dear ScreenTab-User,
```

first of all, thank you for registering. I'm working on ScreenTab since about three years now. Your donation encourages me to develope and improve ScreenTab in the future.

With this archiv, you get the other PlugIn-Modules, which are personalized with your address (you see it in the Information-Requester). Please DON'T SPREAD these modules !

If I release new modules, I will inform you by Email. You can get them for free by Email or for a small fee by disk.

```
                  Installation

                DirWalker-PlugIn

                LoadEject-PlugIn
               --
```


1.2 Installation

Installation

Attention: to use these modules you must have installed ScreenTab V3.1 or

higher ! If you haven't it already you can find the latest version on AmiNet in the directory "util/cdity".

Just copy the files with the ".plugin" extension to the directory, which you have chosen during installation of ScreenTab as destination for the PlugIn-Modules. You can activate them like the others via the preferences program.

The DirWalker-PlugIn needs the FileID-Library. If necessary, copy it from the "libs"-Directory to your "LIBS:"-Assign. It also uses the ReqTools-Library which isn't included in this archive but should be installed on every Amiga system.

If available you can copy the language catalogs of the directory "Catalogs/<language>" to "LOCALE:Catalogs/<language>/ScreenTab/PlugIns" (where <language> is your native language).

1.3 PlugIn: DirWalker

DirWalker

With the DirWalker-PlugIn you can easily browse through the file tree of a device or a directory. When the PlugIn gets to an entry which isn't a directory, it examines the file and opens a user defined function menu for this special file type (or a default menu).

The files are scanned by the FileID-Library which I have included in this archive. This library returns a special ID value for each file. You need this value to define the actions for the file type. I've added a small tool called 'GetFileID' to this archive. With this program you can easily get the ID value of a file.

The parameters for the PlugIn:

ROOT - With this parameter you specify the location where the DirWalker shall start. This can be a physical or logical drive or a directory.

Examples: HD0:, Workbench:, Workbench:C

ACTIONPREFS - The file type actions are defined in a separate data file. Here you must enter the full path to this file. Please refer to the description for more information.

IGNOREINFO - If you specify this option, all Icon-Files (.info) will be ignored.

ADDMISCMENU - If this option is active then the default menu will be added to each action menu as a sub-entry.

USEIMAGES - ScreenTab tries to connect an image to each file (if specified in the ActionPrefs).

1.4 Description of the ActionPrefs-File

Format of the ActionPrefs

The ActionPrefs-File is a normal text file which defines the contents of the function menu for specific file types. For each recognized file type you must enter a definition like the following:

```
[40
Show Amiga-Guide;C:AmigaGuide {f}
]
```

In the first line, you can see an opening square bracket and a number. The bracket marks the beginning of a new file type. The number is the ID value returned from the FileID-Library. You can enter more ID values by separating them with a semicolon (i. e. for different picture types). A carriage return is necessary after this line.

The second line defines the only entry in this action menu. The first part until the semicolon is the text which appears in the menu. The second part specifies the command which will be executed. The {f}-Option tells the PlugIn to insert the selected file here so that the executed command line could look like that:

```
C:AmigaGuide "HELP:Installer.guide"
```

Now a little more complex example:

```
; Pictures (19 = IFF, 97 = JPEG, 56 = GIF)
[19;97;56
Show picture;SYS:Utilities/Visage {f}
]
```

All lines which aren't between bracket pairs are ignored. So you can easily enter some comments like the above. As you can see in the second line that we have specified three different file types which will all be handled with this menu.

Because it isn't possible to define all types of files, you can enter a default menu. This menu will be shown for all files which have no type definition. Simply do this by not entering an ID value, i. e.:

```
; This is the default menu
[
Show me ...;C:More {f}
Rename ...;C:Rename {f} {s;New name ?}
]
```

If you specify the option ADDMISCMENU in the parameter string then the default menu will be added to each action menu as a sub-entry. If you want to give this sub-entry another name than "Misc actions" then type the wished name after the opening bracket, i. e.:

```
; This is the default menu
[Some other nice actions
Show me ...;C:More {f}
Rename ...;C:Rename {f} {s;New name ?}
]
```

I think you now understand how the whole thing works. To enable other actions there are other {X}-Options which you can enter in the command part of the action lines:

```
 {f}

{f-}

{s;X}

{n;X}

{d;X}

{p;X}

{q;X;Y}
```

Usage of patterns

Although the FileID-Library actually knows a lot of file types, it can't identify special textfiles such as source codes or HTML-Documents. But these files often have a unique extension: C-Source mostly ".c", HTML-Docs ".html" or program documentation ".doc".

You can now use these extensions to connect an action menu to a file. All you have to do is to insert a new line in the menu definition which begins with the string "%P=" and continues with an AmigaDOS-Pattern:

```
%P=#?.html
```

In this case all files with the extension ".html" will be connected to the action menu which contains this string.

You can also combine the ID values and the %P-Option. So the DirWalker first tries to identify the file with the IDs and then, if it wasn't successful, compares the file name with the pattern of the %P-Option.

If you can't enter an ID value and still want use the pattern matching then you MUST add the dummy value 0 as the ID. If you don't do this the default menu will be replaced:

```
; HTML-Files
[0
%P=#?.(html|htm)
Open browser;Work:Internet/IBrowseDemo/IBrowse file:///{f-}
]
```

Usage of images

Since version 2.0 of DirWalker you can also use images in the action menus (you have to specify the option USEIMAGES in the parameter string). Like with the %P-Option you must enter a new, unique line in the action menu:

```
%I=Globe
```

The string after the equals sign is the filename of the image you want to use for this file type. Here it's important that you have set the tool-type IMAGEDIR of ScreenTab to the directory which contains the images you normally use (also read the chapter "Important notes" in the ScreenTab-Docs). If you haven't set this tool-type you must enter the FULL path of the image file !

To boldly go where no Start-Button has gone before you can also display images in front of the entries of each action menu. Just add a semicolon to the entry followed by the filename of the image:

```
Open browser;Work:Internet/IBrowseDemo/IBrowse file:///{f-};Globe
```

A complete definition for an action menu for HTML-Documents could then look so:

```
; HTML-Files
[0
%I=Globe
%P=#?.(html|htm)
Open browser;Work:Internet/IBrowseDemo/IBrowse file:///{f-};Globe
]
```

Finally, some important notes

- if you want to use the '{'-Character (i. e. 'C:Echo ${Workbench}) then put a backslash '\' in front of it: 'C:Echo $\{Workbench}'. You only have to do it for the start bracket, not for the end bracket !

- {X}-Options may not be nested

- each action entry needs a new line

- the ending bracket must be entered in a new line

- the DirWalker uses a relatively simple parser so don't try any extra-ordinary experiments :)

1.5 Options for action entries

Options for action entries

{f} - This is the selected file with its complete path in quotation marks.

{f-} - This is the selected file without quotation marks. Please use this option only if it's necessary (that's when {f} doesn't

```
          work !). I actually use it only for HTML-Documents which I
          want to open with the WWW-Browser.

{s;X}   - This one asks the user for a string. You can i. e. demand a new
          name for a file which shall be renamed. X is the title of the
          requester.
          Example:
          Rename ...;C:Rename {f} {s;Rename the file to ...}

{n;X}   - The same like {s} but the user must enter here a number.

{d;X}   - This option opens a file requester where the user must select
          a directory. X is the title of the file requester. So you can
          easily copy a file to a destination with the following line:
          Copy to ...;C:Copy {f} {d;Destination}

{p;X}   - This option has nearly the same effect as 'd', but here the
          DirWalker asks for a complete file path. So if we take a look
          at the above example, we can copy a file to a destination
          directory with a new file name:
          Copy to as ...;C:Copy {f} {p;Destination file}

{q;X;Y} - With this option you can easily ask for confirmation of an
          action. X stands for the question text and Y for the answers.
          So if you want to have confirmation before deleting a file,
          add the line:
          Delete ...;C:Delete {f} {q;Really ?;Yes|No}

          The answers must be separated by a '|' and the left answer must
          be the positive one. If the user selects 'No' in this case,
          the whole command won't be executed.

I've included an example ActionPrefs-File in this archive.  Please  have  a
look at it, if you need more help.
```

1.6 PlugIn: LoadEject

```
LoadEject

Some devices (i. e. CD-ROM drives) have the  possibility  to  eject  and/or
load  a  medium  by program instruction. If you don't specify the ONLYEJECT
option the PlugIn works like a toggle gadget which first opens  the  device
then the next time it closes the device and so on.

Parameters:

DEVICE      - device which will get the message
              Examples: cdrom.device, telmexatapi.device ...

UNIT        - unit of the device

ONLYEJECT   - if this option is specified, the PlugIn only sends "Eject"-
              Messages to the device. This may be useful for Zip-Drives
              or even normal floppy drives if supported.
```

1.7 Einleitung

English
Hallo ScreenTab-Benutzer,

erstmal vielen Dank für die Registrierung. Ich arbeite an ScreenTab nun schon seit etwa drei Jahren. Ihr Beitrag spornt mich an, ScreenTab weiterzuentwickeln und noch zu verbessern.

Wie versprochen, erhalten Sie hiermit die anderen, bisher entwickelten PlugIn-Module. Diese Module wurden mit Ihren persönlichen Daten versehen (werden im Info-Requester des PlugIns angezeigt). Bitte geben Sie diese Module NICHT weiter !

Sollten neue Module erscheinen, bekommen Sie diese auf Wunsch kostenlos (Email) oder gegen eine Unkostengebühr auf Diskette zugeschickt.

Installation

DirWalker-PlugIn

LoadEject-PlugIn
--

1.8 Installation

Installation

Achtung: wenn Sie diese Module benutzen wollen, muß die Version 3.1 oder höher von ScreenTab installiert sein. Falls diese bei Ihnen noch nicht installiert ist, dann finden Sie die aktuellste Version auf dem AmiNet im Verzeichnis "util/cdity".

Kopieren Sie die Dateien mit der Endung ".plugin" in das Verzeichnis, das Sie bei der Installation von ScreenTab als Zielverzeichnis für die PlugIn-Module angegeben haben. Sie können Sie dann wie die anderen über das Voreinstellungsprogramm aktivieren.

Das DirWalker-Modul benötigt die FileID-Library. Wenn Sie auf Ihrem System noch nicht installiert ist, kopieren Sie sie aus dem Verzeichnis "libs" in das "LIBS:"-Assign Ihres Systems. Darüberhinaus muss auch noch die ReqTools-Library installiert sein, die aber eigentlich auf jedem System vorhanden sein sollte (falls nicht, ist sie auf dem Aminet zu finden).

Um die deutschen Versionen der Module nutzen zu können, kopieren Sie die

Dateien mit den Endungen ".catalog" aus dem Verzeichnis "Catalogs/deutsch" ins Verzeichnis "LOCALE:Catalogs/deutsch/ScreenTab/PlugIns".

1.9 PlugIn: DirWalker

DirWalker

Mit dem DirWalker-PlugIn können Sie sehr einfach durch den Dateibaum eines Laufwerks oder Verzeichnisses navigieren. Wenn das PlugIn auf einen Eintrag stösst, der kein Verzeichnis ist, untersucht es die Datei und zeigt ein benutzerdefiniertes Funktionsmenü an, das von dem erkannten Dateityp abhängig ist. Falls eine unbekannte Datei auftaucht, kann ein Standard-Menü angezeigt werden.

Die Dateien werden mit Hilfe der FileID-Library untersucht, die ich diesem Archiv beigefügt habe. Diese Library liefert eine Identifikationsnummer für jede Datei. Diesen Wert brauchen Sie für die Definition der Funktionsmenüs. Mit dem kleinen Shell-Tool 'GetFileID' können Sie sehr einfach die ID-Nummer einer Datei ermitteln.

Hier die Parameter für das PlugIn:

ROOT - Mit diesem Parameter geben Sie an, wo der DirWalker startet. Dies kann ein physisches oder logisches Laufwerk oder ein Verzeichnis sein:

Beispiele: HD0:, Workbench:, Workbench:C

ACTIONPREFS - Die Funktionsmenüs für die verschiedenen Dateitypen werden in einer separaten Datei gespeichert. Mit diesem Parameter müssen Sie den Pfad zu dieser Datei angeben. Bitte lesen Sie auch das Kapitel
Beschreibung ActionPrefs
.

IGNOREINFO - Wenn Sie diesen Parameter angeben, werden alle Icon-Dateien (.info) ignoriert.

ADDMISCMENU - Wenn dieser Parameter angegeben wird, wird das definierte Default-Menu jedem Funktionsmenü als Untereintrag angehängt

USEIMAGES - ScreenTab ordnet, soweit in der ActionPrefs-Datei angegeben, allen Dateien ein Image zu.

1.10 Beschreibung der ActionPrefs-Datei

Format der ActionPrefs-Datei

Die ActionPrefs-Datei ist ein normales Textfile, das die Funktionsmenüs für die verschiedenen Dateitypen definiert. Für jeden zu erkennenden Dateitypen

muss eine Definition angelegt werden, die beispielsweise folgendermaßen aussehen könnte:

```
[40
Show Amiga-Guide;C:AmigaGuide {f}
]
```

In der ersten Zeile sehen Sie eine öffnende, eckige Klammer und eine Zahl. Die Klammer markiert den Anfang einer neuen Funktionsmenü-Definition. Die Zahl ist die Identifikationsnummer, die von der FileID-Library zurückgegeben wird. Sie können hier auch mehrere IDs durch Semikolon getrennt angeben (z. B. für verschiedene Bildformate). Nach den IDs muss ein Zeilenvorschub kommen.

Die zweite Zeile definiert den einzigen Eintrag dieses Funktionsmenüs. Der erste Teil bis zum Semikolon zeigt den Text, der im Menü erscheint. Im zweiten Teil wird der Befehl angegeben, der ausgeführt werden soll, wenn dieser Eintrag angewählt wird. Die {f}-Option teilt dem PlugIn mit, dass es hier den Namen der angewählten Datei eintragen soll. So könnte die letztendlich ausgeführte Befehlszeile folgendermassen aussehen:

```
C:AmigaGuide "HELP:Installer.guide"
```

Jetzt wird’s ein bisschen komplizierter:

```
; Pictures (19 = IFF, 97 = JPEG, 56 = GIF)
[19;97;56
Show picture;SYS:Utilities/Visage {f}
]
```

Alle Zeilen, die nicht zwischen dem Klammernpaar stehen, werden ignoriert. So können Sie einfach ein paar Kommentare einfügen. In der zweiten Zeile können Sie sehen, dass dieses Funktionsmenü für drei Bild-Dateitypen aufgerufen wird, die alle vom gleichen Programm angezeigt werden sollen.

Natürlich ist es nicht möglich, für alle existierenden Dateitypen Funktionsmenüs zu definieren. Deshalb gibt es die Möglichkeit, ein Default-Menü anzugeben, das für unbekannte Dateien verwendet wird. Dazu lassen Sie bei der Definition einfach den ID-Wert weg, z. B.:

```
; This is the default menu
[
Show me ...;C:More {f}
Rename ...;C:Rename {f} {s;New name ?}
]
```

Wenn Sie die Option ADDMISCMENU im Parameterstring angeben, wird jedem Funktionsmenü, das Sie definieren, das Default-Menu als Zusatzeintrag angefügt. Soll dieser Untereintrag einen anderen Namen bekommen als "Misc actions", können Sie nach der öffnenden Klammer den Titel dieses Eintrages angeben, also z. B.:

```
; This is the default menu
[Some other nice actions
Show me ...;C:More {f}
Rename ...;C:Rename {f} {s;New name ?}
]
```

Nun müsste eigentlich klar sein, wie der ganze Mechanismus funktioniert. Es gibt noch andere {X}-Optionen, die Sie als Teil des Kommandos verwenden können:

{f}

{f-}

{s;X}

{n;X}

{d;X}

{p;X}

{q;X;Y}

Benutzung von Mustern

Obwohl die FileID-Library sehr viele Dateitypen kennt, kann sie spezielle Arten von Text-Dateien (z. B. Quellcodes oder HTML-Dokumente) nicht eindeutig erkennen. Diese Dateien haben jedoch meistens eine "Extension" (der Teil nach dem Punkt), die sie identifizieren: C-Quellcodes beispielsweise haben meistens die Erweiterung ".c", HTML-Dokumente ".html" oder normale Anleitungsdateien ".doc".

Diese Erweiterungen können auch für die Zuordnung der Funktionsmenüs benutzt werden. Dazu tragen Sie in die Definition des Dateityps für HTML-Dokumente eine Zeile ein, die z. B. so aussieht:

```
%P=#?.html
```

Die Zeichenkette nach dem Gleichheitszeichen ist ein AmigaDOS-Pattern, wie Sie es vielleicht schon aus der Anwendungsdefinition von ScreenTab kennen. Damit werden alle Dateien, die mit dem String ".html" enden, diesem Funktionsmenü zugeordnet.

Sie können die ID-Werte und die %P-Option auch kombinieren. Dann wird zuerst versucht, die Datei anhand der ID-Werte zu erkennen und erst anschließend mit dem Muster verglichen.

Wenn Sie keinen ID-Wert angeben können und trotzdem die Mustererkennung nutzen wollen, müssen Sie als ID-Wert eine 0 eintragen, da ansonsten das Default-Menü ersetzt wird:

```
; HTML-Files
[0
%P=#?.(html|htm)
Browser öffnen;Work:Internet/IBrowseDemo/IBrowse file:///{f-}
]
```

Benutzung von Images

Seit der Version 2.0 von DirWalker können in den Funktionsmenüs auch Images

benutzt werden (dabei muß die Option USEIMAGES im Parameterstring angegeben werden). Ähnlich wie bei der Musterbenutzung muß dazu für den Dateityp eine (einmalige) Zeile eingefügt werden, z. B.:

```
%I=Globe
```

Die Zeichenkette nach dem Gleichheitszeichen gibt den Dateinamen des Images an, das für diesen Dateityp angezeigt werden soll. Bitte beachten Sie hier, daß der Tool-Type IMAGEDIR von ScreenTab das Verzeichnis enthalten sollte, in dem sich die Images befinden, die Sie normalerweise benutzen (siehe auch "Wichtige Hinweise" in der ScreenTab-Doku). Wenn dieser Tool-Type NICHT gesetzt ist, muß der Pfad zur Image-Datei vollständig angegeben werden !

Wenn Sie noch ein Stück weiter gehen wollen, können Sie auch jedem Eintrag im Funktionsmenü eines Dateityps ein Image zuordnen. Dazu ergänzen Sie den Eintrag einfach am Ende mit einem weiteren Semikolon und dem Namen der Imagedatei, z. B.:

```
Browser öffnen;Work:Internet/IBrowseDemo/IBrowse file:///{f-};Globe
```

Eine vollständige Definition eines Funktionsmenüs für HTML-Dokumente könnte also folgendermaßen aussehen:

```
; HTML-Files
[0
%I=Globe
%P=#?.(html|htm)
Browser öffnen;Work:Internet/IBrowseDemo/IBrowse file:///{f-};Globe
]
```

Einige Hinweise zum Schluß

- falls Sie die geschweifte Klammer in ihren Ausdrücken brauchen (z. B. 'C:Echo ${Workbench}'), dann setzen Sie einen Backslash '\' vor die öffnende Klammer: 'C:Echo $\{Workbench}'. Vor die schliessende Klammer muss kein Backslash gesetzt werden !

- {X}-Optionen können nicht geschachtelt werden

- jeder Funktionseintrag benötigt eine neue Zeile

- die Abschlussklammer muss in einer separaten Zeile stehen

- der DirWalker benutzt einen relativ simplen Parser, deshalb bitte keine aussergewöhnlichen Experimente :)

1.11 Optionen für Funktionseinträge

Optionen für Funktionseinträge

{f} - Diese ist die angewählte Datei mit Pfad in Anführungszeichen gesetzt.

{f-} - Diese ist die angewählte Datei ohne Anführungszeichen. Bitte verwenden Sie diese Option nur dann, wenn Sie wirklich notwendig ist (d. h. wenn {f} nicht funktioniert !). Der einzige Fall, bei dem ich sie benutzen mußte trat bei der Behandlung von HTML-Dokumenten auf, die ich mit einem WWW-Browser öffnen wollte.

{s;X} - Diese fragt den Benutzer nach einem String. Sie können so beispielsweise nach dem neuen Namen einer Datei fragen, die umbenannt werden soll. X steht hier für den Fragetext.

Beispiel:
Rename ...;C:Rename {f} {s;Rename the file to ...}

{n;X} - Dasselbe wie {s}, allerdings muss der Benutzer hier eine Nummer eingeben.

{d;X} - Bei dieser Option öffnet DirWalker einen Verzeichnisrequester, in dem der Benutzer ein Verzeichnis anwählen muss. Hier steht der X-Parameter für den Titel des Requesters. So können Sie einfach eine Datei in ein beliebiges Verzeichnis kopieren:

Copy to ...;C:Copy {f} {d;Destination}

{p;X} - Diese Option hat fast den gleichen Effekt wie 'd', allerdings muß der Benutzer hier einen kompletten Dateipfad angeben. Mit dem obigen Beispiel könnten Sie eine Datei unter einem neuen Namen in ein anderes Verzeichnis kopieren:

Copy to as ...;C:Copy {f} {p;Destination file}

{q;X;Y} - Mit dieser Option können Sie den Benutzer nach einer Bestätigung fragen. X steht dabei für den Fragetext und Y für die Antworten. Wenn Sie also eine Bestätigung haben wollen, bevor eine Datei gelöscht wird, schreiben Sie einfach:

Delete ...;C:Delete {f} {q;Really ?;Yes|No}

Die Antworten müssen mit einem '|' getrennt sein und die erste Antwort muss der positive Fall sein. Wenn hier der Benutzer 'Nein' anklickt, wird der ganze Befehl nicht ausgeführt.

Ich habe diesem Archiv ein Beispiel für eine ActionPrefs-Datei hinzugefügt. Bitte schauen Sie auch dort nochmal nach, wenn etwas unklar ist.

1.12 PlugIn: LoadEject

LoadEject

Manche Geräte (z. B. CD-ROM-Laufwerke) bieten die Möglichkeit, das Medium per Programm auswerfen bzw. einziehen zu lassen. Wenn Sie hier die Option ONLYEJECT nicht angeben, funktioniert das PlugIn wie ein Kippschalter. Zuerst wird das Gerät (Device) geöffnet (eject) und beim nächsten Mal wieder geschlossen (load) usw. .

Parameter:

```
DEVICE      - angeschlossenes Device, das die Nachrichten bekommen soll
              Beispiele: cdrom.device, telmexatapi.device ...

UNIT        - die Einheit (Unit) des Devices

ONLYEJECT   - wenn diese Option angegeben wird, werden nur Nachrichten zum
              Auswurf des Mediums angeschickt. Dies kann z. B. bei Zip-
              Laufwerken oder sogar bei normalen Diskettenlaufwerken
              (soweit unterstützt) nützlich sein.
```