Scripts for PCManFM

PCMan File Manager (PCManFM) is a file manager designed to be a lightweight and fast replacement for Nautilus or Konqueror, for example. It aims to be a usable file manager (and nothing more) and follows the Freedesktop.org standards. For basic usage and on which platforms the program is available read more about here.

Path locations

Build-in actions, desktop applets and default MIME are stored in:

- `/usr/share/file-manager/actions`
- `/usr/share/applications`
- `/usr/share/applications/defaults.list`

The corrospending user-space for actions, desktop applets and default MIME are stored in:

- `/userdata/system/.local/share/file-manager/actions/`
- `/userdata/system/.local/share/applications`
- `/userdata/system/.config/mimeapps.list`

MIME types

Unlike Windows systems, it is much less common in Linux to identify files by the 'file extensions'. Instead, the operating system recognises the file type based on file permissions and metadata, not just the extension. However, common file extensions are still used for organisation and compatibility, for example for compressed archives such as .gz and .bz2 or Office formats such as .odt and .docx or for some special usage like `make`

So yes, we are talking about MIME (aka Multipurpose Internet Mail Extensions) types, rather then file extenions. Best praxis to determine the MIME-type that PCManFM uses is, if you just right-click the file and then read its properties. Or you can just select the file and you'll the file-description into the down-left corner of the filemanager. Valid descriptions are `plain/text` or `inode/directory` (for directories) or `application/zip` - you've got the idea.

Link a MIME type to an application (manual)

Doubleclick on a file that is not yet bound to an application, then a small window appears that's asking which application should be used open the file. You click on `Custom Command Line` register and from there you can type the command or browse your filesystem to select a valid file-application.

Important steps:

1. Fill: Application name (free selectable, as you like)
2. Tick: Set selected application as default for this file type

All data is set to: `/userdata/system/.config/mimeapps.list`, if you edit this file you will see an entry like

```
[Default Applications]
font/ttf=userapp-l3afpad-UMKSE3.desktop
```

So you see: Associated MIME is **font/ttf** and used application is **l3afpad**.

The **.desktop**-file contains all the additional data and can be found in `/userdata/system/.local/share/applications` from here you can change some values or set your own script in the **EXE**-Section and see even the **Name** you gave it. **Type=Application** is important here, this will indicate PCManFM that this file is an application type.

```
[Desktop Entry]
Type=Application
Name=Open TTF
Exec=l3afpad %f
Categories=Other;
NoDisplay=true
MimeType=font/ttf
Terminal=false
```

PCManFM special descriptors

PCManFM and its desktop-files support several parameters, most common is the one for the **EXEC** section which is essential for file handling and file operation.

Parameter	Description	Form
%b	(first) basename	singular
%B	space-separated list of basenames	plural
%c	count of selected items	irrelevant
%d	(first) base directory	singular
%D	space-separated list of base directory of each selected items	plural
%f	(first) file name	singular
%F	space-separated list of selected file names	plural
%h	hostname of the (first) URI	irrelevant
%m	mimetype of the (first) selected item	singular
%M	space-separated list of the mimetypes of the selected items	plural
%n	username of the (first) URI	irrelevant
%s	scheme of the (first) URI	irrelevant
%u	(first) URI	singular
%U	space-separated list of selected URIs	plural
%w	(first) basename without the extension	singular
%W	space-separated list of basenames without their extension	plural
%x	(first) extension	singular
%X	space-separated list of extensions	plural
%%	the "%" character	irrelevant

Action Defintions

A custom action is a “desktop entry” inside the directory `/userdata/system/.local/share/file-manager/actions` or `/usr/share/file-manager/actions`. A desktop entry has a name like **NAME.desktop** (“NAME” is sometimes called its “ID”). It starts with the `[Desktop Entry]` group, which can contain the following keys and their corresponding values:

Key	Description	Necessary
Type	This can be Action or Menu. The latter creates a sub-menu with more actions	Yes
Name	The label of the action as it appears in the context menu - localized variant can be used like Name[fr]	Yes
Icon	The name of a theme icon	No
Description	A free description of the action	No
Enabled	A user might define many actions or menus and choose to only enable some of them from time to time.	No
Profiles	The ordered list of the profiles attached to this action like profile-1;profile-2;	Yes

Profile Defintion

A profile tells the action what to do. Each profile that is added to the Profiles key must be defined in a `[X-Action-Profile profile-id]` group, where **profile-id** is its identifying string, like `[X-Action-Profile profile-zero]`. The group can have the following keys:

Key	Description	Necessary
Name	Not really needed and defaults to empty. It is just a convenience.	No
Exec	The command to execute possibly with arguments. Parameters are often used with this key.	Yes
MimeTypes	The MIME type(s) list with which this action appears. Each mimetype may be fully specified (e.g. audio/mpeg;) or as a group (e.g. image/*;). Mimetypes may be negated (e.g. audio/*;!audio/mpeg;)	Yes
Basenames	A list of base directory names the selection should match for this profile to appear. The * character is accepted as a wildcard. Base names may be negated (e.g. *;!*.h;)	No
Folders	A list of paths the current base directory must be in for the action to appear. A folder path may be negated (e.g. /data;!/data/resources/secret;)	No
SelectionCount	Whether this profile may be selected depending of the count of the selection. This is a string of the form ''< or = or > NUMBER''. Examples of valid strings are: ''=1 or >1 or <10 ''(do not forget the = sign after the key).	No

Special MIME types

Some of well-known mimetypes include:

- `all/all`: matches all items
- `all/allfiles`: matches only files
- `inode/directory`: matches only directories

PCManFM creating an entry in Contextmenu and using Cascaded Contextmenus

Set an entry as Contextmenu

If you place your **.desktop** files to `/userdata/system/.local/share/file-manager/actions/` you can create entries in context menus by right clicking on a file or folder. I've created a small application that calculates the md5sum of all selected files. Talking about the Type you see here the **Type=Action** is present. You'll also need the profile settings.

system.md5sum.desktop

```
[Desktop Entry]
Type=Action
Name=Show md5 Checksum
Profiles=Desktop

[X-Action-Profile Desktop]
MimeTypes=all/allfiles
Exec=bash -c "md5sum %F | yad --text-info"
```

Creating a Cascaded Contextmenu

Again, just change the **Type** you want to descripe into the **.desktop** file and use **Type=Menu** - rather undocumented it seems not to work to set the Name only instead use **Tooltip** to get the entry visible into the context menu. Then just enter the **.desktop**-files you want into your **ItemList** (remember, only the one placed in the DIR `action` will work) and just dismiss the desktop-extension.

menu.cascade-example.desktop

```
[Desktop Entry]
Type=Menu
Icon=menu
Name=
Tooltip=★ Misc. Actions >
ItemsList=md5sum;further-action-1;further-action-2;
```

Scripting Example

File Toggle to set filenames to upper/lower case

Here you see how PCManFMs internal parametes are used. %d is the path (dirname) of the first file, so change directory to this. Later %B is used to work with the filenames only to deal with the for-loop and

to operate the filenames. The big challange is to mask all quotes. Selection Count is set to hinder unintended renaming (for example you marked 1000 of rom files and hit the context entry for example).

Destination: `/userdata/system/.local/share/file-manager/actions/`

system.toggle-filenames.desktop

```
[Desktop Entry]
Type=Action
Name=Toggle filenames upper-/lowercase
Profiles=Batocera
SelectionCount=<50

[X-Action-Profile Batocera]
MimeTypes=all/allfiles
Exec=bash -c "cd %d; [[ %b =~ [A-Z] ]] && { for f in %B; do mv \"$f\" \"${f,,}\"; done; } ||  { for f in %B; do mv \"$f\" \"${f^^}\"; done; }"
```

Create WINE autorun.cmd

You enter the gamefolder and click on a windows-exe-file. From there select `Create autorun.cmd` (Attention, no backup of your old `autorun.cmd` will be made). I made this because here you see that you heavily have to look out which variables you are using. If you need a literal `%` you have to write it twice - escape all quotes and use `printf` rather then `echo`. This script will work only if you are within `/userdata/roms/windows`.

- p1 = Location of all wine-games
- p2 = Directory of our current EXE
- p3 = Windows game directoy itself

So with clever subtrating the strings you can extract all data you need and you are able to write back the needed data to `autorun.cmd`

Destination: `/userdata/system/.local/share/file-manager/actions/`

wine.create-autorun.desktop

```
[Desktop Entry]
Type=Action
Name=Create autorun.cmd
Folders=/userdata/roms/windows/*;
Profiles=Batocera
SelectionCount==1

[X-Action-Profile Batocera]
MimeTypes=application/x-ms-dos-executable
Exec=bash -c "p1=/userdata/roms/windows/; p2=%d; p3=${p2/$p1/};
```

```
p3=${p3%%%%\/*}; printf 'CMD=\"%%s\"\nDIR=%%s' %b \"${p2/$p1$p3/.}\" > \"$p1$p3/autorun.cmd\";"
```