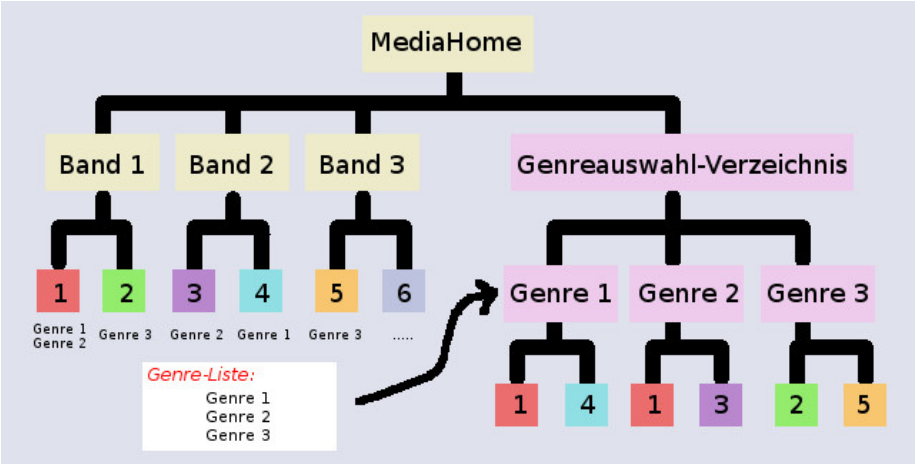
Anwendungsinformationen zu "raise-genres.sh"

Prinzip:

In einem **Genreauswahl-Verzeichnis** werden für jeden in einer **Genre-Liste** enthaltenen Eintrag **Genre-Verzeichnisse** erstellt.

In den Genre-Verzeichnissen werden **symbolische Links auf die Medien-Verzeichnisse** angelegt, die in **Genre-Informations-Dateien** entsprechende Zuordnungen zu Genres in der **Genre-Liste** enthalten.

Graphisches Beispiel: Musik-Alben (1 bis 6) in Band-Verzeichnissen werden Genres zugeordnet ...



Verwendung:

`raise-genres.sh` *`Konfiguration`*

Konfiguration ist der Name eines im Konfigurationspfad vorhandenen Files. Ohne Argument ausgeführt (oder wenn die angegebene Konfiguration nicht existiert), werden vorhandene Konfigurationen angezeigt.

Installation:

Das Skript "raise-genres.sh" systemweit in `/usr/local/bin` oder lokal in `~/bin` entpacken. Konfigurationsdateien anlegen (beliebiges Verzeichnis, am Besten verstecktes Verzeichnis im Home-Verzeichnis) oder Beispiel-Konfigurations-Dateien entpacken.

Konfiguration:

Im Skript "raise-genres.sh" ist der Pfad des Verzeichnisses anzugeben, in dem die Konfigurationsdateien abgelegt werden (Config=*Pfad*).

Eine Konfiguration ist die Auflistung von Optionen in einem Textfile. Die Dateinamen sind beliebig, sollten aber möglichst kurz und sprechend gewählt werden. Um eine bestimmte Konfiguration zu verwenden, wird der Dateiname dem Skript als Argument übergeben (*z.B.* die Konfigurationsdatei "album" wird aufgerufen mit dem Kommando `raise-genres.sh album`).

Zur Konfiguration stehen folgende Optionen zur Verfügung (rot verpflichtend, grün optional):

MediaHome ist der Pfad des Basisordners. Von diesem ausgehend werden alle Pfade angegeben. Er muss vorhanden sein.

GenreVerz ist der Name des Genreauswahl-Verzeichnisses. Ist das Verzeichnis nicht vorhanden, wird es als Unterverzeichnis im Basisordner angelegt (mehrgliedrige Pfade sind nicht möglich!).

TagName ist der Name der Genre-Informationsdateien (*z.B.* TagName=.atag).

TagPfad ist der Pfad der Genre-Informations-Dateien und kann aus Verzeichnisnamen und/oder Wildcards bestehen. Am Ende steht immer ${TagName}, die Datei nach der gesucht wird.
(*z.B.* */*/${TagName} findet "Nils Petter Molvær/1998 - Khmer/.atag")

Genre ist der Pfad der Genre-Liste. Für die darin enthaltenen Genres werden Verzeichnisse angelegt.

Bilder ist der Pfad des Folder-Icon-Verzeichnisses. Wenn Bezeichnungen der darin enthaltenen Bilddateien mit angelegten Genres übereinstimmen, werden sie als "folder.jpg" in die Genre-Verzeichnisse kopiert. Ein Bild mit dem Namen des Genreauswahl-Verzeichnisses wird in dieses als "folder.jpg" kopiert.

ExcludeString ist ein Regulärer Ausdruck, der auf Verzeichnisse passt, die NICHT durchsucht werden sollen. Auszuschließen sind vor allem angelegte Genreauswahl-Verzeichnisse, wo sonst eventuell auch Pfade zu Genre-Informationen gefunden werden könnten – in der Übung sind Verzeichnisse ausgeschlossen, die mit einer Ziffer beginnen, gefolgt von einem Leerzeichen (2 Musik-Alben nach Genre).

IncludeString ist ein Regulärer Ausdruck, der auf Verzeichnisse passt, die durchsucht werden sollen, wenn kein ExcludeString angegeben wurde. Andere Verzeichnisse werden nicht durchsucht.

append=ja deaktiviert das Löschen aller Dateien im Genreauswahl-Verzeichnis. Es werden auch keine neuen Strukturen angelegt. Wenn Genres in den angegebenen Dateien mit den Genre-Informationen (TagName=) auf die Namen der bestehenden Verzeichnisse passen (Jazz → 2 Jazz), werden Links hinzugefügt. Es können dadurch Files in einer anderen Verzeichnistiefe oder mit anderes bezeichneten Genre-Informations-Dateien hinzugefügt werden.
Wenn "append=ja", sind die Pfade für die Genre-Liste (Genre=) und die Folder-Icons (Bilder=) ohne Wirkung. Es wird nur bestehenden Strukturen hinzugefügt!

verwendeTopVerz=ja aktiviert die Ausgabe des übergeordneten Verzeichnisnamens in runden Klammern.
Beispiel: das Pfadelement "Ane Brun/2005 - Duets/" erzeugt einen Link "2005 - Duets (Ane Brun)".

holeTopTop=ja aktiviert die Ausgabe des "über-übergeordneten" Verzeichnisses in runden Klammern, wenn "verwendeTopVerz=ja" aktiviert ist.
Beispiel: das Pfadelement "Ane Brun/Live-Alben/2007 - Live In Scandinavia/" erzeugt wieder einen Link "2007 - Live In Scandinavia (Ane Brun)".

reverse=ja aktiviert die umgekehrte Ausgabe des Link-Namens – die Bezeichnung des übergeordneten Verzeichnisses (oder des diesem übergeordneten Verzeichnisses) wird an erster Stelle genannt und der Name des die Dateien enthaltenden Verzeichnisses in runden Klammern.

Nur wirksam, wenn "verwendeTopVerz=ja".
Beispiel: das Pfadelement "Ane Brun/2005 - Duets/" erzeugt einen Link "Ane Brun (2005 - Duets)".

TitelExString ist ein Regulärer Ausdruck, der auf Verzeichnisse zutrifft, deren Name nicht verwendet werden soll (es wird aber trotzdem ein Link erzeugt). Der Linkname besteht dann nur aus der Bezeichnung des übergeordneten Verzeichnisses (oder des diesem übergeordneten Verzeichnisses).
Nur wirksam, wenn "verwendeTopVerz=ja" und "reverse=ja".
Beispiel: TitelExString='^Singles' bewirkt bei Pfadelement "Ane Brun/Singles/" → "Ane Brun"

CutJahrString ist ein Regulärer Ausdruck, der für ein Datum am Beginn des Verzeichnisnamens steht - das Datum wird entfernt und die Jahreszahl in runden Klammern hinzufügt.

PlaylistString ist ein Regulärer Ausdruck, der auf Erweiterungen von Dateien passt, die für Playlists verwendet werden sollen. Am Ende steht immer $, da nur Dateien berücksichtigt werden sollen, bei denen diese Zeichenfolge am Ende des Strings steht.
Beispiel: (flac|mp3|m4a)$
Es werden hier noch keine Playlists erzeugt, sondern Pfadlisten, die in Folge für Playlists verwendet werden können (siehe unten).

IFS ist fest im Skript konfiguriert (anders als in der Übung!). In der Konfigurationsdatei könnte ein abweichender Wert gesetzt werden.

Beispiel: Meine reale Konfiguration "chron" für Musik-Alben nach Genres:

Das im Skript festgelegte Verzeichnis für die Konfigurationen ist
Config=~/.scripts/raise-genres/conf

In diesem Verzeichnis befindet sich u.a. die Konfigurationsdatei **~/.scripts/raise-genres/conf/chron**

```
### Konfiguration: chronologisch ###
MediaHome="/mnt/archive/cont/Audio Music"
GenreVerz="2 Albums by Genre"
TagName=.atag
TagPfad="*/*/*/${TagName}"
Genre=( `cat ~/.scripts/raise-genres/Genre-List_chron` )
Bilder=~/.scripts/raise-genres/icons
ExcludeString=
IncludeString='^[[:upper:]]{1,3}'
verwendeTopVerz=ja
holeTopTop=nein
append=nein
reverse=nein
CutJahrString=
TitelExString=
PlaylistString='flac|mp3|m4a$'
```

Erläuterungen zu dieser konkreten Konfiguration:

Die Genre-Informations-Dateien liegen in beliebigen Unterverzeichnissen von Unterverzeichnissen von Unterverzeichnissen von MediaHome.
Beispiel: A/Ane Brun/2015 - When I'm Free/.atag

Pfade werden nicht ausgeschlossen, sondern nur bestimmte, mit 1 bis 3 Großbuchstaben beginnende Pfade eingeschlossen. Das sind bei mir "Anfangsbuchstaben-Verzeichnisse", die auch mehrere seltenere Buchstaben zusammenfassen.
Beispiel: XYZ/Zero 7/2001 - Simple Things/.atag

Nicht verwendete Optionen können auch auskommentiert (# ...), leer- oder ganz weggelassen werden. Es können beliebig viele Konfigurationsdateien nebeneinander existieren.

"RandomPlaylist.sh"

Prinzip:

Mit den von raise-genres.sh erzeugten "Playlist"-Pfadlisten werden unter Zuhilfenahme von $RANDOM zufällig ausgewählte Pfade aneinandergereiht.

Es ist vorgesehen, das Skript automatisch, jedes Mal vor dem Start der Multimedia-Software, auf dem Gerät auszuführen, auf dem die Playlists verwendet werden sollen. Bei mir wird das Skript ausgeführt, bevor KODI gestartet wird.
Danach kann das Programm auch manuell aufgerufen werden, um neue Playlists zu erzeugen.

Verwendung:

`RandomPlaylist.sh [`*`Titelzahl`*`]`
Titelzahl ist der Anzahl der pro Genre erzeugten Playlisteinträge. Ohne Argument ausgeführt, wird die konfigurierte Default-Titelzahl verwendet (siehe Konfiguration).

Installation:

Das Skript "RandomPlaylist.sh" systemweit in `/usr/local/bin` oder lokal in `~/bin` entpacken.

Konfiguration:

Die Konfiguration wird direkt im Skript "RandomPlaylist.sh" vorgenommen. Zur Konfiguration gibt es folgende Optionen:

Playlists ist der Pfad des Verzeichnisses, in dem die Playlists abgelegt werden. Wenn Playlists mit absoluten Pfaden benutzt werden, ist der Ablageort beliebig. Der Pfad muss aber vorhanden sein.

MediaHome ist der Pfad des Basisordners aus dem Skript "raise-genres.sh". Der Pfad muss auf dem System, auf dem das Skript ausgeführt wird, vorhanden sein.

GenreVerzArray ist ein Array, dessen Elemente Namen von Genreauswahl-Verzeichnissen sind, die Playlist-Informationen enthalten. Das ist der Fall, wenn *z.B.* PlaylistString='(flac|mp3|m4a)$' in der "raise-genre.sh"-Konfiguration angegeben ist. Das Array kann ein oder mehrere Elemente enthalten.
Beispiel: GenreVerzArray=("2 Albums by Genre" "4 Videos")

Ext ist die für die Playlists verwendete Dateinamen-Erweiterung (*Empfehlung:* Ext=.m3u).

Titelzahl ist die Default-Anzahl der erzeugten Playlist-Einträge (*z.B.* Titelzahl=50).

Es gibt noch eine zusätzliche Option, falls die Playlists nicht auf dem System erzeugt werden können, auf dem sie verwendet werden sollen. Fertige Multimedia-Geräte erlauben häufig nicht das Ausführen eigener Shell-Scripts.

AltHome ist ein alternativer Pfad des Basisordners, der in den Playlists anstelle von MediaHome verwendet wird. AltHome kann auch ein relativer Pfad, ausgehend vom Verzeichnis mit den Playlists, sein.

Beispiel: Playlists="/mnt/archive/cont/Audio Music/0 Playlists"
MediaHome="/mnt/archive/cont/Audio Music"
AltHome="../.."
(Playlist-Dateien befinden sich dann in "$MediaHome/0 Playlists/4 Videos" und "../.." ist der relative Pfad zurück zu MediaHome)

"tag-album.sh"

Prinzip:

Eine Folge von interaktiven Eingaben wird als einfache Liste in Textdateien in Unterverzeichnisse **des aktuellen Verzeichnisses** geschrieben. Die so erzeugten oder erweiterten Textfiles stehen als Genre-Informations-Dateien für die Auswertung durch "raise-genres.sh" zur Verfügung.

Verwendung:

`tag-album.sh [e|n [von [bis]]]`
e fügt bestehenden Dateien weitere Genres hinzu (default), ***n*** schreibt die Datei neu.
von ist die das erste zur Kennzeichnung verwendete Verzeichnis, ***bis*** das letzte.
Alle Argumente sind optional, ohne 1. Argument oder mit jedem anderen Argument als ***n*** werden die Genres bestehenden Dateien hinzugefügt.

Installation:

Das Skript "tag-album.sh" systemweit in `/usr/local/bin` oder lokal in `~/bin` entpacken.

Konfiguration:

Die Konfiguration wird direkt im Skript "tag-album.sh" vorgenommen. Es gibt einen einzigen Konfigurationspunkt.

Tagname ist der Name der Genre-Informations-Dateien, die angelegt werden.

Skript ausführen:

Das Skript muss immer in dem Verzeichnis ausgeführt werden, in dessen Unterverzeichnisse geschrieben werden soll!

Dieses Skript ist als Hilfsmittel gedacht, wenn mit der Kategorisierung eines Musik-Archivs begonnen wird. Danach wird man die Files eher einzeln anlegen und das Skript nur sehr selten benötigen.

Es gibt natürlich auch Alternativen zu "raise-genres.sh". Multimedia-Systeme haben meist eine eigene Dateiverwaltung und viele auch "smarte" Playlists, die jeweils zu erkunden wären. Die Bibliothek der Multimedia-Suite KODI bietet eine Fülle an Möglichkeiten.

Verwaltung von Multimedia-Dateien mit NFO-Files (für KODI)

In Verzeichnissen mit Multimedia-Dateien werden Textdateien mit der Dateinamenerweiterung .nfo abgelegt. Die Textdateien tragen den gleichen Namen wie die Multimediadateien. Wenn sich nur ein Multimedia-File im Verzeichnis befindet, ist die Benennung egal.

Der Inhalt der Datei kann ein Web-Link auf eine Medienbibliothek sein, oder alle Informationen direkt darin angegeben haben. KODI sieht die Benutzung lokaler Informationen nur vor, wenn im Netz kein Informationen automatisch gefunden werden. Will man Zuordnungen nach eigenen Vorstellungen treffen können, gibt es aber keine Alternative zu lokalen Informationen.

Die lokalen Informationen werden im XML-Format bereitgestellt. Für "Artist", "Genre", "Year", auch "Path", sowie vieles mehr gibt es definierte Felder, denen ein Wert zugewiesen werden kann. Auch die mehrfache Verwendung eines Feldes ist möglich, zum Beispiel für mehrere Genres.

Es können "Smart Playlists" definiert werden, deren Regeln die zufällige Zusammenstellung einer Playlist bestimmen. Diese Regeln umfassen alle Arten von Operatoren, wie "gleich", "entält", "größer", "kleiner", "nicht", und können alle in den NFO-Dateien vorhandenen Felder auswerten.

Wer sich im Detail dafür interessiert findet alle näheren Informationen im KODI-Wiki: http://kodi.wiki/view/NFO_files

Die Möglichkeiten sind beinahe unendlich, aber der Verwaltungsaufwand steht dem nicht nach - jedenfalls dann nicht, wenn man sich auf eigene Zuordnungen stützen will. Und das wollte ich!

Das Paket "raise.genres.zip"

Das Paket "raise.genres.zip" enthält außer den Skripten noch meine Original-Konfigurationsdateien. Für die eigene Verwendung müssen diese entsprechend angepasst werden.

Die enthaltenen Dateien "vid" und "vfull" enthalten keine Konfiguration, sondern Aufrufe von raise-genres.sh mit je zwei versteckten Konfigurationen. So können mit einem Aufruf Multimedia-Dateien auf zwei Verzeichnisebenen berücksichtigt werden.

Zusätzlich enthält das Paket noch "**uRandomPlaylist.sh**" - Funktionalität und Konfiguration entspricht "RandomPlaylist.sh". Die (Pseudo-)Zufallszahlen werden im Unterschied nicht mit $RANDOM, sondern mit dem vom Kernel generierten Zufalls-Bit-Strom /dev/urandom erzeugt. Das sollte eine noch "zufälligere" Auswahl zur Folge haben. Das Skript kann alternativ zu RandomPlaylist.sh verwendet werden.

Über Feedback freue ich mich: kontakt@ernstlx.com