Musik und Figuren (Work in Progress)

Musik

Etwas unpraktisch ist, dass die Ordner/Dateinamen nur aus fortlaufend numerierten 2/3 stelligen Ziffern bestehen sollen. Die Erweiterung dieser Benennung ist sehr fehleranfällig.

Anbei ein Python-Skript, dass alle Ordner und Dateien in einen temporären Ordner kopiert und in das SD-Karten kompatible Format umbenennt . Damit kann mann den SD-Karten Inhalt am PC mit langen Dateinamen verwalten und bei Bedarf einfach einen Export für den Tonuino generieren.

```
"""
Developed and tested under Linux. Should work in Windows as well.

Use this script to export an arbitrary named mp3 folder structure to Tonuino SD-card.
Requirements to mp3 folder structure:
    - requires "advert" and "mp3" folder for sd card
    - maximal directory depth 1, i.e. no subfolders
All not mp3 files are ignored for the export.

Basic usage:
-------------

$ python to_tonuino.py

Plan for copying files:
/home/user/data/unterhaltung/musik/tonuino_sdCard/250424_dev/01_party/aSong.mp3 ->
/tmp/250502-1901_tonuino_sdCard_952_zqjt/01/001.mp3
/home/user/data/unterhaltung/musik/tonuino_sdCard/250424_dev/01_party/bSong.mp3 ->
/tmp/250502-1901_tonuino_sdCard_952_zqjt/01/002.mp3
/home/user/data/unterhaltung/musik/tonuino_sdCard/250424_dev/02_ambiente/ambiente1.mp3 ->
/tmp/250502-1901_tonuino_sdCard_952_zqjt/02/001.mp3
```

```
/home/user/data/unterhaltung/musik/tonuino_sdCard/250424_dev/02_ambiente/ambiente2.mp3 ->
/tmp/250502-1901_tonuino_sdCard_952_zqjt/02/002.mp3
/home/user/data/unterhaltung/musik/tonuino_sdCard/250424_dev/advert -> /tmp/250502-
1901_tonuino_sdCard_952_zqjt/advert
/home/user/data/unterhaltung/musik/tonuino_sdCard/250424_dev/mp3 -> /tmp/250502-
1901_tonuino_sdCard_952_zqjt/mp3

Playlist:
01: 01_party
001.mp3: aSong.mp3
002.mp3: bSong.mp3
02: 02_ambiente
001.mp3: ambiente1.mp3
002.mp3: ambiente2.mp3

Copying files ...
Finished copying files to /tmp/250502-1901_tonuino_sdCard_952_zqjt



Advanced usage with command line arguments:
-------------------------------------------

For testing call as:
$ python to_tonuino.py /run/media/user/EAGET --test
where /run/media/user/EAGET is your sdCard export-folder.

If you like the generated copy plan, use
$ python to_tonuino.py /run/media/user/EAGET
for actual export.
"""

import os
import shutil
from pathlib import Path
import tempfile
from typing import List, Tuple
from datetime import datetime
import argparse
```

```
SKIP_DIRS = ["advert", "mp3"]



def build_mp3_copy_plan(source_dir: Path, target_dir: Path, skip_dirs: List[str]) ->
List[Tuple[Path, Path]]:
    plan = []
    # dirs = [item for item in source_dir.iterdir() if item.is_dir()]
    dirs = sorted([item for item in source_dir.iterdir() if item.is_dir()])
    for dd, dir in enumerate(dirs):
        if dir.name in skip_dirs:
            continue
        fns = sorted(dir.glob('*.mp3'))
        for ff, fn in enumerate(fns):
            fp_from = dir / fn
            fp_to = target_dir / f"{dd+1:02}" / f"{ff+1:03}.mp3"
            plan.append((fp_from, fp_to))
    return plan

def print_playlist(plan: List[Tuple[Path, Path]]):

    playlist = ""
    dir_to_prev = None;
    for fp_from, fp_to in plan:
        if dir_to_prev != fp_to.parent.name:
            dir_to_prev = fp_to.parent.name
            playlist += f"{fp_to.parent.name}: {fp_from.parent.name}\n"
        playlist += f"\t{fp_to.name}: {fp_from.name}\n"
    return playlist[:-1]





if __name__ == "__main__":

    parser = argparse.ArgumentParser(description="Process an optional filepath.")
    parser.add_argument(
        "target_dir",
        nargs="?",
        type=str,
        default=None,  # default value if not provided
```

```
        help="Where to export sd Card content to."
    )
    parser.add_argument(
        "--test",  # Flag name
        action="store_true",  # Action to take (sets it to True if present)
        help="If testmode: do not copy any files and do not create any folders."
    )
    args = parser.parse_args()
    target_dir = args.target_dir
    mode_test = args.test




    skip_dirs = SKIP_DIRS
    source_dir = Path.cwd()
    if target_dir is None:
        target_dir = tempfile.mkdtemp(prefix=f"{datetime.now().strftime('%y%m%d-
%H%M')}_tonuino_sdCard_")
    target_dir = Path(target_dir)
    plan = build_mp3_copy_plan(source_dir, target_dir, skip_dirs=skip_dirs)

    print("\nPlan for copying files:")
    for fp_src, fp_dst in plan:
        print(f"\t{fp_src} -> {fp_dst}")
    for dir in skip_dirs:
        print(f"\t{source_dir/dir} -> {target_dir/dir}")

    playlist = print_playlist(plan)
    print(f"\nPlaylist:\n{playlist}")

    if mode_test:
        print("\n" + "Test mode on: no copying of files".center(50, "=") + "\n")
        exit(0)

    print("\nCopying files ...")
    for dir in skip_dirs:
        shutil.copytree(source_dir / dir, target_dir / dir)
    for fp_src, fp_dst in plan:
        os.makedirs(os.path.dirname(fp_dst), exist_ok=True)
        shutil.copy2(fp_src, fp_dst)
```

```
    print(f"Finished copying files to {target_dir}")
```

Figuren

Der Tonuino akzeptiert Tags in verschiedenen Formaten. Ich persönlich habe schon getestet: NATG215 und MIFARE Classic 1K. Ein paar Ideen, um Kindertaugliche Datenträger zu erstellen sind:

- RFID Karten mit schönen Aufklebern bekleben.
- RFID Coins (kleine nicht klebende Plättchen) kaufen und Spielfiguren (z.B. Ü-Ei) daraufkleben.
- RFID Aufkleber auf Spielzeug kleben, z.B. Duplo Männchen.
- Ntag 216 RFID Implantate (Achtung auf 13,56 MHz achten, nicht 13,... kHz), Größe ca. 12mm x 2mm in Plastiktiere mit Lötkolben einschmelzen.

Folgende Erfahrungen haben **WIR ALLE** mit den verschiedenen Konzepten gemacht. (Bitte tragt hier gerne eure Erfahrungen mit ein.

Konzept	Alter Kind	Erfahrung	Ansprechpartner
RFID Coins (kleine nicht klebende Plättchen) kaufen und Spielfiguren (z.B. Ü-Ei) daraufkleben.	1 Jahr	Verschluckgefahr falls sich RFID Coin ablöst.	Markus Dutschke
RFID Aufkleber auf Spielzeug kleben, z.B. Duplo Männchen.			
Ntag 216 RFID Implantate (Achtung auf 13,56 MHz achten, nicht 13,... kHz), Größe ca. 12mm x 2mm in Plastiktiere mit Lötkolben einschmelzen.			

Revision #6
Created 2025-04-23 16:51:33 UTC by Markus Dutschke
Updated 2025-05-02 17:31:16 UTC by Markus Dutschke