File Organizer Tool

Narzędzie CLI do rekurencyjnej organizacji, deduplikacji i konsolidacji plików z wielu katalogów źródłowych do jednego katalogu docelowego.

Wymagania

- Python 3.6+
- System Linux
- Tylko biblioteki standardowe (brak zewnętrznych zależności)

Instalacja

```
git clone https://github.com/jszwagie/File-organizer-tool.git
cd File-organizer-tool
```

Użycie

Składnia

```
python3 main.py <katalog_docelowy> <katalog_źródłowy1> [katalog_źródłowy2
...] [--config ŚCIEŻKA]
```

Argumenty

Argument	Opis
`katalog_docelowy`	Katalog docelowy (X), gdzie zostaną skonsolidowane wszystkie pliki
`katalog_źródłowy`	Jeden lub więcej katalogów źródłowych (Y1, Y2, ...) do przeszukania
`--config`	Ścieżka do pliku konfiguracyjnego (domyślnie: `~/.clean_files`)

Podstawowe wywołanie

```
python3 main.py /home/user/Documents /home/user/Downloads /home/user/Backup
```

Z własnym plikiem konfiguracyjnym

```
python3 main.py /target /source1 /source2 --config ./my_config.ini
```

Konfiguracja

Program wymaga pliku konfiguracyjnego. Domyślnie szuka `~/.clean_files`, następnie `.clean_files` w bieżącym katalogu.

Format pliku konfiguracyjnego

```
[Settings]
# Domyślne uprawnienia plików (format octal)
default_permissions = 644

# Znaki zabronione w nazwach plików (rozdzielone spacjami)
bad_chars = : ; * ? $ # \ ' "

# Znak zastępujący zabronione znaki
replacement_char = _

# Rozszerzenia plików tymczasowych (rozdzielone spacjami)
temp_extensions = .tmp .bak .swp .~ .log
```

Parametry konfiguracji

Parametr	Opis	Przykład
`default_permissions`	Docelowe uprawnienia plików w formacie octal	`644` (rw-r--r--)
`bad_chars`	Znaki do zastąpienia w nazwach plików	`: ; * ? $ # \ ' "`
`replacement_char`	Znak zastępujący niedozwolone znaki	`_`
`temp_extensions`	Rozszerzenia plików tymczasowych do usunięcia	`.tmp .bak .swp .~ .log`

Fazy działania

Program działa w 4 kolejnych fazach, każda z ponownym skanowaniem struktury katalogów:

FAZA 1: Sanityzacja

- **RENAME**: Zmiana nazw plików zawierających zabronione znaki
- **CHMOD**: Normalizacja uprawnień do wartości z konfiguracji

FAZA 2: Usuwanie śmieci

- **DELETE**: Usunięcie pustych plików (0 bajtów)
- **DELETE**: Usunięcie plików tymczasowych (zgodnie z `temp_extensions`)

FAZA 3: Deduplikacja i wersjonowanie

- **DELETE**: Usunięcie duplikatów (pliki o identycznej zawartości) - zachowuje najstarszy
- **MOVE**: Przeniesienie oryginału do katalogu docelowego (jeśli nie jest tam)
- **DELETE**: Usunięcie starszych wersji plików o tej samej nazwie - zachowuje najnowszy

FAZA 4: Konsolidacja

- **MOVE**: Przeniesienie unikalnych plików z katalogów źródłowych do docelowego

Interakcja z użytkownikiem

Dla każdej sugerowanej akcji program wyświetla:

```
FILE: /ścieżka/do/pliku
ISSUE: Opis problemu
ACTION: TYP_AKCJI -> cel (jeśli dotyczy)
Confirm: [y]es, [n]o, [a]lways for this type, [q]uit:
```

Opcje odpowiedzi

Klawisz	Działanie
y	Wykonaj tę akcję
n	Pomiń tę akcję
a	Wykonaj tę i wszystkie następne akcje tego samego typu
q	Zakończ program

Przykłady

Przykład 1: Podstawowe porządkowanie

```
# Porządkowanie plików z Downloads i Desktop do Documents
python3 main.py ~/Documents ~/Downloads ~/Desktop
```

Przykład 2: Konsolidacja kopii zapasowych

```
# Zebranie plików z wielu backupów do jednego miejsca
python3 main.py /mnt/backup/consolidated /mnt/backup/2023 /mnt/backup/2024
/mnt/old_drive
```

Przykład 3: Z własną konfiguracją

```

```

```
# Użycie własnego pliku konfiguracyjnego
python3 main.py /target /source --config /etc/clean_files.conf
```

Testowanie

Generator struktury testowej

Projekt zawiera skrypt do generowania struktury testowej:

```
# Wygeneruj środowisko testowe
python3 test_structure_generator.py

# Wygeneruj w konkretnym miejscu
python3 test_structure_generator.py /tmp/test_env

# Tylko wyświetl drzewo istniejącego katalogu
python3 test_structure_generator.py --tree
python3 test_structure_generator.py /tmp/test_env -t
```

Uruchomienie testu

```
# 1. Wygeneruj strukturę testową
python3 test_structure_generator.py

# 2. Uruchom program na strukturze testowej
python3 main.py test_env/target_X test_env/source_Y1 test_env/source_Y2

# 3. Sprawdź wynik
python3 test_structure_generator.py --tree
```

Struktura testowa zawiera

- Pliki puste (0 bajtów)
- Pliki tymczasowe (.tmp, .bak, .log, .swp)
- Pliki ze złymi znakami w nazwie (: ; ? $ # ' ")
- Pliki z niestandarwymi uprawnieniami (777, 755, 666)
- Duplikaty (identyczna zawartość, różne nazwy/lokalizacje)
- Wersje plików (ta sama nazwa, różna zawartość)
- Pliki unikalne do konsolidacji

Struktura projektu

```
File-organizer-tool/
├── main.py                         # Główny punkt wejścia
```

```
├── .clean_files                 # Przykładowy plik konfiguracyjny
├── test_structure_generator.py  # Generator struktury testowej
├── README.md                    # Ta dokumentacja
└── src/
    ├── __init__.py              # Eksport modułów
    ├── config.py                # Ładowanie konfiguracji i CLI
    ├── scanner.py               # Skanowanie katalogów
    ├── analyzer.py              # Analiza i generowanie sugestii
    └── executor.py              # Wykonywanie akcji
```

Typy akcji

Akcja	Opis
`DELETE`	Usunięcie pliku
`MOVE`	Przeniesienie pliku do innej lokalizacji
`RENAME`	Zmiana nazwy pliku (w tym samym katalogu)
`CHMOD`	Zmiana uprawnień pliku
`COPY`	Skopiowanie pliku (zachowuje oryginał)
`SKIP`	Pominięcie (brak akcji)

Bezpieczeństwo

- Program jest odporny na spacje i znaki specjalne w nazwach plików
- Każda akcja wymaga potwierdzenia użytkownika (lub trybu "always")
- Operacje są wykonywane sekwencyjnie z możliwością przerwania w dowolnym momencie