

POLITECHNIKA BIAŁOSTOCKA

Wydział Informatyki

Paweł Żukowski

Multifile Renaming Utility

Program narzędziowy do grupowej zmiany nazw plików na podstawie metadanych w nich zawartch

PRACA INŻYNIERSKA

Promotor
dr inż. Marcin Skoczylas

Białystok 2013

Streszczenie

Zażółć gęślą jaźń

Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed diam nonummy nibh euismod tincidunt ut laoreet dolore magna aliquam erat volutpat. Ut wisi enim ad minim veniam, quis nostrud exerci tation ullamcorper suscipit lobortis nisl ut aliquip ex ea commodo consequat. Duis autem vel eum iriure dolor in hendrerit in vulputate velit esse molestie consequat, vel illum dolore eu feugiat nulla facilisis at vero eros et accumsan et iusto odio dignissim qui blandit praesent luptatum zzril delenit augue duis dolore te feugait nulla facilisi. Nam liber tempor cum soluta nobis eleifend option congue nihil imperdiet doming id quod mazim placerat facer possim assum. Typi non habent claritatem insitam; est usus legentis in iis qui facit eorum claritatem. Investigationes demonstraverunt lectores legere me lius quod ii legunt saepius. Claritas est etiam processus dynamicus, qui sequitur mutationem consuetudinum lectorum. Mirum est notare quam littera gothica, quam nunc putamus parum claram, anteposuerit litterarum formas humanitatis per seacula quarta decima et quinta decima. Eodem modo typi, qui nunc nobis videntur parum clari, fiant sollemnes in futurum.

Słowa kluczowe: *Zażółć gęślą jaźń*

Spis treści

Spis treści	3
1 Wstęp	5
1.1 Cel i zakres pracy	6
1.2 Założenia	6
1.3 Plan pracy	6
2 Teoria	7
2.1 Dane w systemie komputerowym	7
2.2 Systemy plików i identyfikacja danych	8
2.2.1 Katalogi i ścieżki do plików	9
2.2.2 Różnice w identyfikacji plików wśród różnych systemów operacyjnych	9
2.3 Metadane zawarte w plikach	10
3 Implementacja	11
3.1 Środowisko pracy	11
3.1.1 System operacyjny FreeBSD	11
3.1.2 Mercurial	11
3.1.3 CMake	11
3.1.4 vim	11
3.2 Wykorzystane biblioteki	11
3.2.1 SigC	12
3.2.2 boost::filesystem	12

3.2.3	boost::property_tree	12
3.3	Rdzeń aplikacji - klasa MruCore	12
3.4	Wyrażenia zawierające metatagi	12
3.5	System modułów i jego implementacja	12
3.6	Typy modułów w MRU	12
3.7	Moduły UI	12
3.7.1	wxWidgetsUi	12
3.7.2	TextUi	12
3.8	Moduły output	12
3.8.1	GenericBoost	12
3.9	Moduły metatagów	12
3.9.1	Count	12
3.9.2	Audio	12
3.9.3	CRC	12
3.9.4	ToLower	12
4	Definicje	13

Rozdział 1.

Wstęp

Z każdym rokiem ludzie oraz same komputery generują coraz większą ilość informacji. Mimo że duża część z nich jest przechowywana w dobrze strukturyzowanych bazach danych, to ciągle, większość ludzi ma bezpośredni dostęp jedynie to tego co przechowuje w systemie plików własnego komputera. <Rodzaje danych, ich zastosowanie>

Od dziesięcioleci dysk twardy pozostaje głównym kontenerem dla danych użytkowników komputerów na całym świecie. <Dane przechowywane w systemach plików>

<Systemy plików, ich cechy wspólne, ograniczenia, metadane>

<problem z identyfikatorami>

1.1 Cel i zakres pracy

Niniejsza praca ma na celu stworzenie programu narzędziowego pozwalającego na automatyczne generowanie identyfikatorów (nazw) plików na podstawie metadanych w nich zawartych. <także danych generowanych przez sam program - CRC np>

Zakres pracy obejmuje:

- Przegląd istniejących rozwiązań - programów i technik wspomagających masową zmianę identyfikatorów plików.
- Porównanie funkcjonalności istniejących narzędzi i ich ograniczeń.
- Projekt oraz implementacja wieloplatformowej architektury modułów.
- Stworzenie parsera wyrażeń zawierających metatagi.
- Projekt graficznego interfejsu użytkownika opartego na bibliotece wxWidgets.
- Implementacja backendu do systemu plików opartego na bibliotece boost::filesystem.
- Implementacja przykładowych modułów metatagów.
- Testy aplikacji.

1.2 Założenia

Gotowa aplikacja powinna być niezależna od systemu operacyjnego w stopniu w jakim pozwalają na to zależności użytych bibliotek. Dzięki modułowej budowie powinna także udostępniać interfejs pozwalający na jej łatwą rozbudowę.

1.3 Plan pracy

<Co w jakim rozdziale się znajduje>

Rozdział 2.

Teoria

W niniejszym rozdziale postaram się przybliżyć obraz problemu identyfikatorów (zwanych również nazwami) plików opisując środowisko i w którym występuje.

2.1 Dane w systemie komputerowym

Jednym z podstawowych elementów systemu komputerowego jest jego pamięć. Od początku istnienia komputerów istniała potrzeba składowania danych wymaganych przy praktycznie każdych operacjach wykonywanych przez jednostkę centralną komputera. Jako że pierwsze systemy komputerowe były wykorzystywane do obliczeń typowo matematycznych, algorytmy na nich uruchamiane nie wymagały wielkich kontenerów na dane. W tych czasach wbudowane rejestry oraz ulotna pamięć RAM zaspokajały potrzeby rynku. Jednak wraz z rozwojem sprzętu i algorytmów na nim uruchamianych pojawiła się potrzeba przechowywania coraz to większej ilości danych jak i (dzięki zastosowaniu architektury von Neumanna) samych programów przez coraz dłuższy czas. Pojawiła się idea nieulotnej oraz pojemnej pamięci - dysku twardego. <!sprawdzić!>

Pojemności pierwszych dysków twardych stanowiły promil dzisiejszych jednostek toteż nie wymagały stosowania systemów plików - były po prostu nieulotnym rozszerzeniem pamięci operacyjnej RAM.

Wraz ze zwiększeniem ich pojemności oraz generalizacją oprogramowania, pojawiła się

potrzeba standaryzowania, kategoryzacji przechowywanych na dyskach danych - tak powstały systemy plików.

??

2.2 Systemy plików i identyfikacja danych

System plików stanowi warstwę abstrakcji między programami, a danymi zapisanymi na nośniku — dysku twardym, karcie pamięci czy też płycie CD. System plików jest metodą zapisu danych — schematem dzięki któremu, programy nie muszą operować na surowych blokach danych lecz mogą korzystać z bardziej wysokopoziomowych deskryptorów plików - węzłów bądź ścieżek dostępu.

Zwykle systemem plików zarządza system operacyjny — to on udostępnia API, a także blokuje lub pozwala na dostęp do danych ze względu na uprawnienia użytkownika, programu lub samego zasobu.

Istnieje wiele typów oraz implementacji systemów plików, które można podzielić na dwie kategorie:

- tradycyjne - znajdujące zastosowanie przy przechowywaniu dowolnych (ogólnych) danych w postaci plików
- specjalne - dostosowane do specyficznych rozwiązań (jak na przykład bazy danych)

Oddzielną kategorię mogą stanowić zdobywające coraz większą popularność wirtualne systemy plików - różnią się one od tradycyjnych i specjalistycznych tym że nie przechowują danych fizycznie na nośniku, a są raczej aplikacjami udostępniającymi (generującymi) struktury danych na żądanie użytkownika/programu. Przykładem takich systemów mogą być: **procfs** - udostępniający dostęp do procesów systemowych i ich atrybutów w systemach rodziny GNU/Linux oraz *BSD, czy też **NFS** (Network File System) — pozwalający na dostęp do systemów plików znajdujących się na innych komputerach w sieci.

2.2.1 Katalogi i ścieżki do plików

Niniejsza praca skupia się na problemie opisywania danych w tradycyjnych systemach plików za pomocą tak zwanych ścieżek do plików.

Tradycyjne systemy plików pozwalają na przechowywanie danych w drzewiastej strukturze danych zwanej drzewem katalogów. W większości implementacji każdy węzeł takiego drzewa może być katalogiem, plikiem lub dowiązaniem do innego węzła. Dodatkowo węzły katalogów jako jedyne mogą posiadać węzły podległe — podkatalogi. Każdy węzeł prócz węzła-korzenia jest identyfikowany przez unikalny względem węzła-rodzica identyfikator zwany nazwą pliku/katalogu.

Warto zauważyć iż struktura drzewa katalogów nie wymusza sposobu rozkładu danych w systemie plików — tak długo jak identyfikatory pozostają unikalne, pliki przez nie opisywane mogą znajdować się w tym samym katalogu¹.

2.2.2 Różnice w identyfikacji plików wśród różnych systemów operacyjnych

Format ścieżki do pliku narzucany jest niezależnie od zastosowanego systemu plików przez system operacyjny.

Systemy kompatybilne ze standardem POSIX, wywodzące się z Unixów takie jak Apple MacOS czy rodzina BSD, a także rodzina GNU/Linux używają drzew katalogów z pojedynczym, nienazwanym korzeniem oznaczanym symbolem prawego ukośnika (slash) — '/'.

Symbol prawego ukośnika jest również używany jako separator elementów (poziomów) ścieżki i nie może stanowić elementu identyfikatora węzła w wymienionych środowiskach.

Przykład ścieżki zgodnej ze standardem POSIX:

`/home/idlecode/projects/mru/doc/main.tex`

¹W praktyce ilość plików które mogą należeć do jednego węzła zależy od rozmiaru licznika użytego w implementacji.

Systemy operacyjne z rodziny Windows korporacji Microsoft² wykorzystują natomiast lewy ukośnik (backslash) — '\' — jako separator komponentów ścieżki oraz uniemożliwiają stosowanie większej ilości symboli w nazwach.

System plików systemu Windows może posiadać kilka korzeni (po jednym dla każdego wolumenu/dysku) oznaczanych pojedynczymi, zwykle dużymi literami alfabetu łacińskiego. Litera dysku wraz z symbolem dwukropka poprzedza właściwą ścieżkę do pliku.

Przykład ścieżki używanej w systemach operacyjnych Windows korporacji Microsoft:

```
C:\Users\idlecode\My Documents\Projects\MRU\doc\main.tex
```

Ze względów ogólnie stosowanego API (języka C), w przypadku obu³ wyżej wymienionych schematów nazwy elementów nie mogą zawierać znaku NUL (o kodzie heksadecymalnym 0x00), który jest interpretowany jako koniec łańcucha znaków.

Istnieje jeszcze kilka schematów zapisu ścieżek, które nie zostały przybliżone ze względu na zakres niniejszej pracy.

<!TODO: zachowywanie wielkości znaków i ich interpretacja!> <!TODO: kodowanie używane do zapisu ścieżki!> <!TODO: maksymalna długość ścieżki i nazwy pliku!>

2.3 Metadane zawarte w plikach

²Istnieje wiele więcej systemów operacyjnych używających podobnego schematu

³System MacOS nie posiada tego ograniczenia

Rozdział 3.

Implementacja

3.1 Środowisko pracy

Użyty język programowania; toolchain; system operacyjny; kontrola wersji;

3.1.1 System operacyjny FreeBSD

3.1.2 Mercurial

3.1.3 CMake

3.1.4 vim

3.2 Wykorzystane biblioteki

asd

- 3.2.1 SigC
- 3.2.2 boost::filesystem
- 3.2.3 boost::property_tree
- 3.3 Rdzeń aplikacji - klasa MruCore
- 3.4 Wyrażenia zawierające metatagi
- 3.5 System modułów i jego implementacja
- 3.6 Typy modułów w MRU
- 3.7 Moduły UI
 - 3.7.1 wxWidgetsUi
 - 3.7.2 TextUi
- 3.8 Moduły output
 - 3.8.1 GenericBoost
- 3.9 Moduły metatagów
 - 3.9.1 Count
 - 3.9.2 Audio
 - 3.9.3 CRC
 - 3.9.4 ToLower

Rozdział 4.

Definicje

\mathbb{R} - zbiór liczb rzeczywistych

I - macierz opisująca obraz wejściowy

w - szerokość obrazu wejściowego, liczba kolumn macierzy I

Lista publikacji

C. Zet, H. Kieseewetter, M. Skoczylas, L. Westerberg, R. Spohr. A system for irradiating polymer films with a preset number of ions. GSI Scientific Report, Gesellschaft für Schwerionenforschung mbH (GSI), 154, 2003

C. Zet, C. Foslau, M. Skoczylas, L. Westerberg, R. Spohr. System for irradiating polymer films with a preset number of ions. SIELMEN, 4th International Conference on Electromechanical and Power Systems vol. 2, 167-170, 2003

M. Skoczylas, K. Andrzejewski. Krzyżtopór. Telewizja Polska, wyd. Rzeczpospolita, Akademia Filmu i Telewizji Warszawa, 2005