Deutsche Nationalbibliothek

6. November 2024

Inhaltsverzeichnis

Abkürzungsverzeichnis

AEN Automatische Erschließungsverfahren; Netzpublikationen

DNB Deutsche Nationalbibliothek

1 Einführung

Das Projekt `pica-rs` ermöglicht eine effiziente Verarbeitung von bibliografischen Metadaten, die in PICA+, dem internen Format des OCLC-Katalogsystems, kodiert sind. Das Kommandozeilenprogramm `pica` stellt unterschiedliche Kommandos zur Verfügung, um Daten auszuwählen, statistisch aufzubereiten oder für die Weiterverarbeitung in *Data Science*-Frameworks wie Polars (Python) oder der Programmiersprache R nutzbar zu machen. Die Anwendung ist in der Programmiersprache *Rust* geschrieben und lässt sich unter den Betriebsystemen Linux, macOS und Windows verwenden. Die Kommandos lassen sich über die Standard-Datenströme miteinander verketten, wodurch sich leicht Metadaten-Workflows erstellen und automatisieren lassen.

Die Entwicklung von `pica-rs` wurde vom Referat Automatische Erschließungsverfahren; Netzpublikationen (AEN) der Deutsche Nationalbibliothek (DNB) initiiert und wird dort für die Erstellung von Datenanalysen sowie für die Automatisierung von Workflows (Datenmanagement) im Rahmen der automatischen Inhaltserschließung genutzt. Weiterhin wird es zur Unterstützung der Forschungsarbeiten im KI-Projekt sowie für diverse andere Datenanalysen innerhalb der DNB eingesetzt.

2 Installation und Einrichtung

Das Kommandozeilen-Tool `pica` lässt sich unter den Betriebssystemen Linux, macOS und Windows nutzen. Folgend wird die Installation von sowie Einrichtung und Konfiguration des Tools beschrieben.

2.1 Installation

2.1.1 Installation unter Linux

Abhängig von der genutzten Linux-Distribution, gibt es unterschiedliche Möglichkeiten Möglichkeiten der Installation. Vorgefertigte Releases stehen auf der Plattform GitHub zum Download bereit[1].

Debian und Ubuntu

Unter Debian GNU/Linux und Ubuntu Linux können fertige `DEB`-Pakete genutzt werden. Diese lassen sich mit dem `dpkg`-Programm wie folgt installiert werden:

```
$ dpkg -i pica_0.25.0-glibc2.35-1_amd64.deb
```

Red Hat, SUSE und CentOS

Für die Distributionen Red Hat Linux, SUSE Linux und CentOS Linux stehen fertige `RPM`-Pakete zum Download bereit. Installieren lassen sich diese mit dem `rpm`-Programm installieren:

```
$ rpm -i pica-0.25.0-glibc2.35-1.x86_64.rpm
```

[1] https://github.com/deutsche-nationalbibliothek/pica-rs/releases

Binary Releases

Soll `pica` nicht über den Paketmanager installiert werden, steht für die Zielarchitektur `x86_64-unknown-linux-gnu` mit den `glibc`-Versionen 2.28, 2.31 und 2.35 fertige *Binary Releases* zur Verfügung[2].

Das `tar`-Archiv enthält neben dem Tool `pica` auch weitere Dateien wie die Shell-Skripte zur Befehlszeilenergänzung:

```
$ tar -tf pica-0.25.0-x86_64-unknown-linux-gnu-glibc2.35.tar.gz
pica-0.25.0-x86_64-unknown-linux-gnu-glibc2.35/
pica-0.25.0-x86_64-unknown-linux-gnu-glibc2.35/pica
pica-0.25.0-x86_64-unknown-linux-gnu-glibc2.35/README.md
pica-0.25.0-x86_64-unknown-linux-gnu-glibc2.35/pica.zsh
pica-0.25.0-x86_64-unknown-linux-gnu-glibc2.35/LICENSE
pica-0.25.0-x86_64-unknown-linux-gnu-glibc2.35/pica.fish
pica-0.25.0-x86_64-unknown-linux-gnu-glibc2.35/pica.bash
```

Eine systemweite Installation von `pica` in das Verzeichnis `/usr/local/bin` kann mit dem `install` erfolgten. Hierfür sind ggf. `root`-Rechte nötig:

```
$ tar xfz pica-0.25.0-x86_64-unknown-linux-gnu-glibc2.35.tar.gz
$ sudo install -m755 pica-0.25.0-x86_64-unknown-linux-gnu-glibc2.35/pica \
    /usr/local/bin/pica
```

2.1.2 Installation unter macOS

Unter *macOS* wird nur die Zielarchitektur `x86_64-apple-darwin` (macOS 10.7+, Lion+) unterstützt. Diese lässt sich analog wie die Linux-Release installieren:

```
$ tar xfz pica-0.25.0-x86_64-apple-darwin.tar.gz
$ install -m755  pica-0.25.0-x86_64-apple-darwin/pica /usr/local/bin/pica
```

2.1.3 Installation unter Windows

Unter Windows (`x86_64-pc-windows-gnu` oder `x86_64-pc-windows-msvc`) kann das Programm direkt dem `zip`-Archiv entnommen werden. Nach einem Wechsel in das Verzeichnis, in dem sich die `pica.exe` befindet, kann das Programm direkt genutzt werden. Soll `pica` aus jedem beliebigen Verzeichnis aus aufrufbar sein, dann muss der Installationspfad in der `PATH`-Umgebungsvariable mit aufgelistet sein.

[2] Die `glibc`-Version des Systems lässt sich mit dem Aufruf `ldd --version` ermitteln.

2.1.4 Aus dem Quellcode installieren

Das Projekt lässt sich auch direkt aus den Quellen kompilieren. Hierfür wird eine aktuelle Rust-Version (>= 1.74.1) mit dem Paketmanager `cargo` benötigt.

Der aktuelle Entwicklungsstand lässt sich wie folgt installieren:

```
$ git clone https://github.com/deutsche-nationalbibliothek/pica-rs.git
$ cd pica-rs
$ cargo build --release
```

Das fertige Release-Binary `pica` liegt im Verzeichnis `target/release/` und kann bspw. in das Verzeichnis `/usr/local/bin` installiert werden:

```
$ install -m755 target/release/pica /usr/local/bin/pica
```

Wenn die Quelle nicht benötigt werden, kann das Projekt auch direkt über den Paketmanager `cargo` installiert werden:

```
$ # Installation der aktuellen Entwicklungsversion
$ cargo install --git https://github.com/deutsche-nationalbibliothek/pica-rs \
    --branch main pica-toolkit

$ # Installation der Version 0.25.0
$ cargo install --git https://github.com/deutsche-nationalbibliothek/pica-rs \
     --tag v0.25.0 pica-toolkit
```

Das fertige Programm befindet sich dann unter `~/.cargo/bin/pica`.

2.2 Einrichtung

2.2.1 Shell

Nach der Installation kann `pica` direkt in einem Kommandozeileninterpreter (*Shell*) verwendet werden. Die Interpretation der Kommandoaufrufe und -optionen ist abhängig von der verwendeten Shell. Alle Beispiele in diesem Manual werden in der bash-Shell unter Linux ausgeführt. Wie die Argumente und Kommandoaufrufe von einer anderen Shell und dem Betriebssystemen interpretiert werden, kann variieren und ist ggf. bei einer Fehlersuche mit einzubeziehen.

Insbesondere unter Windows muss in der PowerShell besondere Vorsicht walten. Hier wird die Zeichenkodierung der Daten bei der Vekettung von Kommandos verändert. Aus diesem Grund wird von der Verwendung der PowerShell abgeraten und stattdessen die Eingabeaufforderung `cmd.exe` empfohlen. Siehe auch den GitHub Issue #371.

Eine wichtige Eigenschaft der verwendeten Shell, die Benutzende immer im Hinterkopf haben sollten, ist die Interpretionen von Zeichenketten. Hier werden ggf. besondere Anweisungen durch die Shell ausgeführt und durch andere Ausdrücke ersetzt. Dieses Verhalten führt zu fehlerhaften `pica`-Anweisungen.

Um in der `bash`-Shell die Fehler zu minimieren, sollten Literale (bspw. Filter- und Selektionsausdrücke) immer in einfachen Anführungszeichen ausgezeichnet werden. Ebenfalls sinnvoll kann die Deaktivierung der History-Substitution mittels `set +H` sein. Dies verhindert das Ersetzen von '`!0`'-Fragmenten durch den letzten Befehl, der von der Shell ausgeführt wurde.

Befehlszeilenergänzung

TODO

2.2.2 Konfiguration

TODO