Betriebssystem-Shells und Command-Line Interfaces

Ein Arbeitsblatt

Einleitung

Für viele ist die grafische Benutzeroberfläche (englisch “graphical user interface”; **GUI**) die einzige Mensch-Maschine-Schnittstelle, die sie kennen. Doch es gibt auch die Befehlszeilenschnittstelle, auf englisch Command-Line Interface (**CLI**). Bei einem CLI geben Benutzer Befehle als Text in einer Shell ein, um das Betriebssystem oder ein Programm zu steuern. Ein CLI ist schwieriger zu erlernen als ein GUI. Das CLI macht es aber einfacher, Software zu konfigurieren sowie Befehle zu wiederholen und Abläufe zu automatisieren. Bestimmte Programme lassen sich nur über ein CLI bedienen. Beispiele solcher Programme sind der SQL-Client `psql` des PostgreSQL-Datenbanksystems oder eben die Betriebssystem-Shells, die in diesem Arbeitsblatt erklärt werden.

Dieses Arbeitsblatt zeigt dir anhand von vier Kapiteln den Umgang mit Betriebssystem-Shells und wie du Umgebungsvariablen anpassen kannst.

Im ersten Kapitel wirst du eine Betriebssystem-Shell starten und ein paar Systembefehle ausführen. Im zweiten Kapitel wirst du lernen, eine zusätzlich installierte Software über ein CLI zu benutzen und zwar anhand des Dateikonverters `ogr2ogr` aus der GDAL/OGR-Programmfamilie. Schliesslich wirst du im dritten Kapitel lernen, wie man Umgebungsvariablen setzt, indem du deinen Prompt anpasst. Zu guter Letzt kannst du im vierten Kapitel dein Wissen anwenden, indem du mit dem Java-Programm `ili2gpkg` Geodaten konvertierst.

Schlüsselwörter: *CLI, Terminal, Konsole, Betriebssystem-Shell (Shell), Prompt, Systembefehl, Verzeichnis, Pfad, Umgebungsvariable*.

Betriebssysteme Windows, macOS und Linux

Dieses Arbeitsblatt behandelt die wichtigsten drei Betriebssysteme, nämlich Windows, macOS und Linux. Darum sind hier die Erläuterungen und Übungen grundsätzlich aufgeteilt in drei Unterkapitel Windows, macOS und Ubuntu. Linux umfasst mehr als 600 Distributionen, welche diverse Unterschiede aufweisen. Wir haben uns bei Linux für Ubuntu entschieden, da es weit verbreitet und einsteigerfreundlich ist.

Dateiverzeichnisse und Pfade

Die drei hier behandelten Betriebssysteme verwalten ihre Dateien in hierarchischen **Verzeichnissen**. In ihnen kannst du als Benutzer navigieren und dir z.B. den Namen und den Inhalt des aktuellen Verzeichnisses ausgeben lassen. Ein Verzeichnis kann Dateien und andere

Verzeichnisse (seine Unterverzeichnisse) enthalten. Ein Verzeichnis, das Verzeichnisse oder Dateien enthält, ist deren Überverzeichnis. Vor allem in GUIs werden Verzeichnisse oft als "Ordner" bezeichnet.

Ein **Pfad** besteht aus einer durch Abgrenzungssymbole (eine Form des Slashes) getrennten Liste von Dateiverzeichnisnamen, die mit einem Verzeichnis- oder Dateinamen endet. Ein Pfad ist eine Adresse im Dateisystem. Mit ihm kann ein Programm, das Betriebssystem und auch der Benutzer Daten orten. Windows verwendet den "Backslash" `\` und macOS und Linux-Distributionen wie Ubuntu den "Slash" `/`.

Es gibt zwei Arten von Pfaden: absolute und relative. Ein **absoluter Pfad** geht immer vom höchsten Verzeichnis im Dateisystem aus. Bei Windows ist dies das Laufwerk, also z.B. `C:` und bei macOS und Ubuntu das Root-Verzeichnis `/`. **Relative Pfade** gehen vom aktuellen Verzeichnis aus. Dies erkennt man daran, dass der Pfad mit einem Punkt `.`, mit zwei Punkten `..` oder direkt mit dem Verzeichnisnamen beginnt. Der Punkt ist ein Platzhalter für das aktuelle Verzeichnis und die zwei Punkte markieren einen Sprung nach oben im Dateiverzeichnis.

Kapitel 1: Umgang mit der Betriebssystem-Shell

In diesem Kapitel wirst du deine Betriebssystem-Shell starten und ein paar Systembefehle ausführen. Und du lernst, wie man Informationen zu einem Systembefehl aufrufen kann.

Jedes der drei Betriebssysteme hat unterschiedliche **Betriebssystem-Shells**. Windows besitzt eine Shell namens "cmd" (offiziell: Windows-Eingabeaufforderung) und eine weitere, die PowerShell. Wir verwenden hier nur die "cmd"-Shell. Das Betriebssystem macOS verwendet die Z Shell als Standard, aber hat auch die Bash Shell vorinstalliert. Unter macOS kann somit in einer sehr ähnlichen Umgebung gearbeitet werden, wie bei Linux-Distributionen, wenn man die Bash Shell verwendet, was im restlichen Verlauf des Arbeitsblattes auch getan wird.

Eine **Shell** ist ein Programm, das die direkte Kommunikation mit dem Betriebssystem ermöglicht. Die Kommunikation findet über ausgeführte Befehle statt. In diesem Arbeitsblatt unterscheiden wir zwischen **Systembefehlen** und Befehlen von zusätzlich installierten Programmen. Jedes Betriebssystem kann andere Namen für bestimmte Befehle, oder sogar mehrere Namen für einen Befehl haben.

Das GUI ist auch eine Shell, da es den Benutzer ebenfalls direkt mit dem Betriebssytem interagieren lässt. Im weiteren Verlauf des Arbeitsblattes wird unter einer Shell nur die CLI-Variante verstanden.

Übung 1: Betriebssystem-Shell und System-Befehle

Zuerst wirst du lernen, wie du deine Betriebssystem-Shell starten kannst. Danach zeigen wir dir ein paar Systembefehle und auch, wie du dich durch Verzeichnisse navigieren kannst. Am Schluss wirst du lernen, wie du Hilfe zu einen Befehl erhältst.

Schritt 1.1 Betriebssystem-Shell starten (Windows)

1. Öffne das Ausführen-Fenster mit `Windowstaste` + `R`.
2. Gib `cmd` ein und klicke **[OK]** oder drücke `Enter`.

Schritt 1.1 Betriebssystem-Shell starten (macOS)

1. Öffne die Spotlight-Suche mit `command` + `Leertaste`.
2. Gib `terminal` ein und drücke `Enter`.
3. Starte Bash mit `bin/bash`, falls sie nicht schon geöffnet ist. Ob Bash läuft, erkennst du daran, dass das Prompt-Symbol (mehr dazu in Kapitel 2) ein Dollarzeichen ist `$`.

Schritt 1.1 Betriebssystem-Shell starten (Ubuntu)

Öffne ein Terminal mit `Ctrl` + `Alt` + `T`.

Schritt 1.2 In Verzeichnissen navigieren — Systembefehle ausführen (Windows)

Nun hast du die Shell von Windows vor dir. Hier kannst du Befehle eintippen und mit `Enter` ausführen lassen. Du kannst die Autovervollständigung durch Drücken von `Tabulator` aktivieren. Das, was du dann schon eingegeben hast, wird vervollständigt und wenn mehrere Möglichkeiten bestehen, kannst du noch mal `Tabulator` drücken und du erhältst die ganze Liste an Möglichkeiten.

Aktuelles Verzeichnis ausgeben und in Vereichnissen navigieren `cd`

Um anzuzeigen, in welchem Verzeichnis du dich momentan befindest, kann man `cd` allein benutzen

Die Hauptfunktion des Befehls ist die Navigation in den Verzeichnissen. Wenn man einen Pfad nach dem Befehl angibt, wechselt man in das entsprechende Verzeichnis. Wenn man anstatt einem Verzeichnis `..` eingibt, geht man einen Ordner nach oben.

Bewege dich zum Desktop. Gib dafür den Befehl `cd Desktop` ein. Wenn du `cd` eintippst und nach dem Leerzeichen den `Tabulator` drückst, kannst du die Autovervollständigung nutzen.

Dateien und Unterverzeichnisse auflisten `dir`

Dieser Befehl gibt eine Lise aller direkten Unterverzeichnisse und Dateien eines Verzeichnisses aus. Wenn man kein Verzeichnis angibt, benutzt es das Verzeichnis, in dem man sich momentan befindet.

Pfade mit Leerzeichen muss man in Anführungszeichen angeben, da ansonsten das CLI das Leerzeichen als Ende des Pfades interpretiert und den Pfad einfach dort abschneidet.

Verzeichnis erstellen `mkdir`

Dieser Befehl erstellt ein Unterverzeichnis. Falls das Verzeichnis bereits besteht, wird eine Fehlermeldung geworfen.

Erstelle nun einen neuen Ordner `test` mit dem Befehl `mkdir test`.

Verzeichnis löschen `rmdir`

Dieser Befehl löscht das danach angegebene Verzeichnis.

Lösche den vorher erstellten Ordner `test` mit `rmdir test`.

Schritt 1.2 Systembefehle ausführen — In Verzeichnissen navigieren (macOS, Ubuntu)

In Verzeichnissen navigieren `cd`

Wenn man einen Pfad nach dem Befehl `cd` angibt, wechselt man in das entsprechende Verzeichnis. Wenn man anstatt einem Verzeichnis `..` eingibt, geht man einen Ordner nach oben.

Bewege dich zum Desktop. Gib dafür den Befehl `cd Desktop` ein. Wenn du `cd` eintippst und nach dem Leerzeichen den `Tabulator` drückst, kannst du die Autovervollständigung nutzen.

Aktuelles Verzeichnis ausgeben `pwd`

Um anzuzeigen, in welchem Verzeichnis du dich momentan befindest, benutze `pwd`. `pwd` steht für "print working directory", was auf deutsch "zeige aktuelles Directory" heisst.

Dateien und Unterverzeichnisse auflisten `ls`

Dieser Befehl gibt alle direkten Unterverzeichnisse und Dateien eines Verzeichnisses zurück. Wenn man kein Verzeichnis angibt, benutzt es das Verzeichnis, in dem man sich momentan befindet.

Pfade muss man, falls sich ein Leerzeichen in ihnen befindet oder sie sehr lang sind, mit Anführungszeichen angeben, da ansonsten das CLI das Leerzeichen als Ende des Pfades interpretiert und den Pfad einfach abschneidet.

Verzeichnis erstellen `mkdir`

Dieser Befehl erstellt ein Unterverzeichnis. Falls das Verzeichnis schon besteht, wird eine Fehlermeldung geworfen.

Erstelle nun einen neuen Ordner `test` mit dem Befehl `mkdir test`.

Verzeichnis löschen `rmdir`

Dieser Befehl löscht das danach angegebene Verzeichnis.

Lösche den vorher erstellten Ordner `test` mit `rmdir test`.

Schritt 1.3 Weitere Systembefehle (Windows)

Benutzernamen ausgeben `whoami`

Der Befehl (Who am I? = Wer bin ich?) gibt den Benutzernamen aus.

Gib `whoami` ein und drücke `Enter`.

Programme als Administrator ausführen

Falls du Administratorrechte auf deinem Benutzer besitzst oder das Passwort eines Administrators kennst, kannst du Programme als Administrator ausführen. Dies ist nötig, um manche Programme zu installieren, Systemänderungen zu vollführen oder bestimme Befehle auszuführen.

Im Command Prompt gibt es keinen Befehl, mit dem man direkt Befehle als Administrator ausführen kann. Stattdessen kann man in der Windows-Ausführfunktion `Windowstaste` + `R` `cmd` eingeben und dann mit `Ctrl` + `Shift` + `Enter` als Administrator ausführen. Oder du kannst auch in der Windows-Suche nach `cmd` suchen und dann **Rechtsklick › Als Administrator ausführen** drücken.

Möglicherweise musst du dich danach als Administrator einloggen oder dein Passwort eingeben.

Wenn dein Prompt nun `C:\Windows\system32>` anzeigt, dann hat dies funktioniert (wenn du den

Prompt nicht verändert hast).

Shell schliessen `exit`

Dieser Befehl schliesst die Shell.

Schritt 1.3 Weitere Systembefehle (macOS, Ubuntu)

Benutzernamen ausgeben `whoami`

Der Befehl (Who am I? = Wer bin ich?) gibt den Benutzernamen aus.

Gib `whoami` ein und drücke `Enter`.

Programme als Administrator ausführen `sudo`

Falls du Administratorrechte auf deinem Benutzer besitzst oder das Passwort des Administrators kennst, kannst du Programme als Administrator ausführen. Dies ist nötig, um manche Programme zu installieren, Systemänderungen zu vollführen oder bestimme Befehle auszuführen.

`sudo` ist ein Bash-Befehl, der den folgenden Befehl als Administrator ausführt. Vor allem wenn man Dateien ausserhalb seines Benutzers verändert, verwendet man ihn sehr oft.

Verwende `sudo -i` um dich als Administrator (in Linux "root" genannt) anzumelden. Alle Befehle, welche du dann ausführst, funktionieren dann genau so, als würde vor jedem Befehl `sudo` stehen. Um dich abzumelden kannst du den Befehl `logout` verwenden.

Shell schliessen `exit`

Dieser Befehl beendet die Shell. Wenn du mehrere Shells geöffnet hast, wird nur die innerste beendet.

Schritt 1.4 Hilfe für einen Systembefehl aufrufen (Windows)

Bei Windows funktioniert dies so: hier wird zuerst der gewünschte Befehl, gefolgt vom Zusatz `/?` eingegeben. Für den Befehl `cd` muss man also `cd /?` benutzen. Man kann auch `help` gefolgt vom Befehl eingeben.

Schritt 1.4 Hilfe für einen Systembefehl aufrufen (macOS, Ubuntu)

Falls man mehr Informationen über einen Befehl erhalten will, kann man in macOS und Ubuntu `man` eingeben, gefolgt vom gewünschten Befehl. Im Beispiel von `pwd` müsste man also im Terminal `man pwd` eingeben.

Danach öffnet sich die Hilfe-Ansicht. Durch diese kannst du mit den Pfeiltasten `↑` oder `↓` scrollen und schliessen mit `Q`.

Nun kennst du die Grundlagen deiner Shell und bist bereit, um mehr kennenzulernen.

Kapitel 2: Zusätzlich installierte Software über ein CLI benutzen

In dieser Übung wirst du Shapefiles in ein GeoPackage konvertieren, und zwar mit der Hilfe der Programmfamilie OGR, die vorgängig zu installieren ist.

Shapefile und GeoPackage

Ein Shapefile ist ein Format für vektorielle Geodaten. Es besteht aus mindestens drei Dateien (immer mit demselben Dateinamen): .shp für die Geometriedaten, .dbf die Sachdaten und .shx als Index der Geometriedaten. Unser Übungsbeispiel enthält auch noch eine weitere Datei .prj, die das Koordinatenreferenzsystem bzw. die Projektion enthält, also zum Beispiel kantonsstrassen_zh.shp, kantonsstrassen_zh.shx, kantonsstrassen_zh.dbf, kantonsstrassen_zh.prj.

Es gibt noch viele weitere Dateien, die je nach Software verwendet werden, was einer der Gründe ist, warum Shapefiles veraltet sind (vgl. Switch from Shapefile).

Ein GeoPackage ist - wie Shapefiles - ein Dateiformat für den Austausch von vektoriellen Geodaten. Es hat einige bessere Eigenschaften als Shapefiles und basiert auf SQLite.

QGIS und OGR

Zur Visualisierung der konvertierten Geodaten kannst du das Desktop-GIS-Programm QGIS verwenden.

OGR steht für "*O*pen*G*IS Simple Features *R*eference Implementation" und ist eine Programm-Familie zur Übersetzung von Vektordaten. OGR ist Teil des Repositories der"*G*eospatial *D*ata *A*bstraction *L*ibrary" (GDAL) enthalten, die für Rasterdaten Übersetzungen benutzt wird. Für diese Übung wirst du aus der OGR-Familie das CLI-Programm "ogr2ogr" verwenden, das Geodaten in andere Formate konvertieren kann.

Installation von QGIS und OGR (Windows)

Für Windows kannst du das OSGeo4W Paket herunterladen. Es enthält nicht nur QGIS, sondern auch ein Terminal, das OGR bereits integriert hat. Achte bei der Installation darauf, dass du auch GDAL installierst.

⇒ Siehe das Arbeitsblatt von OpenSchoolMaps "Einführung in QGIS 3: Einleitung und Vorbereitung" für die Installation von QGIS

Nach der Installation kannst du in der Shell testen, ob OGR installiert wurde.

1. Öffne das Terminal "OSGeo4W Shell".
2. Gib den Befehl `ogrinfo` ein.

Falls das Programm nicht richtig installiert wurde, erhältst du den Output: "Der Befehl "ogrinfo" ist

entweder falsch geschrieben oder konnte nicht gefunden werden."

Installation von QGIS und OGR (macOS)

⇒ Siehe das Arbeitsblatt von OpenSchoolMaps "Einführung in QGIS 3: Einleitung und Vorbereitung" für die Installation von QGIS

1. Zuerst musst du den Paketmanager "Homebrew" herunterladen. Gib dafür `curl -fsSL https://raw.githubusercontent.com/Homebrew/install/HEAD/install.sh` im Terminal ein. (Möglicherweise wirst du danach gebeten, Xcode herunterzuladen. Klicke dann auf **[Installieren]**.)
   `curl` ist ein Shell-Befehl. Er ist zuständig für Datenübertragung und lädt uns Homebrew über die URL herunter. Die Option `-fsSL` (was in Wahrheit vier Optionen sind) hat diese Funktionen:
   - f → Im Falle, dass der Server einen Fehler wirft, soll kein Output gegeben werden.
   - s → Es sollen kein Ladebalken und keine Fehlermeldung anzeigt werden.
   - S → Überschreibt `s` und wirft doch eine Fehlermeldung.
   - L → Der Befehl passt sich an, falls das Package bewegt wurde (auf eine andere URL).
2. Lade mit dem Befehl `brew install gdal` GDAL herunter
   Hier benutzen wir den Homebrew Paket-Manager `brew` mit dem Befehl `install`, also installieren und dem Paketnamen `gdal`.
3. Teste nun ob die Installation erfolgreich lief. Gib den Befehl `ogrinfo` ein.

Falls das Programm nicht richtig installiert wurde, erhältst du den Output: "Der Befehl "ogrinfo" ist entweder falsch geschrieben oder konnte nicht gefunden werden."

Installation QGIS und OGR (Ubuntu)

⇒ Siehe das Arbeitsblatt von OpenSchoolMaps "Einführung in QGIS 3: Einleitung und Vorbereitung" für die Installation von QGIS

1. Zuerst musst du das Repository, also der Ort, an dem sich die benötigten Daten befinden, dem Paket-Manager hinzufügen. Dies geschiet mit dem Befehl `sudo add-apt-repository ppa:ubuntugis/ppa`.
2. Danach kannst du den Paket-Manager updaten, so dass das hinzugefügte Repository eingelesen wird. `sudo apt-get update`
3. Am Schluss kannst du GDAL installieren. `sudo apt-get install gdal-bin`

Nun ist GDAL installiert. Teste ob es richtig installiert wurde, indem du den Befehl `ogrinfo` eingibst. Wenn du keine Fehlermeldung erhalten hast, kannst du nun fortfahren.

Übungsdaten

Bei den Übungsdaten handelt es sich um die Shapefiles der Kantonsstrassen des Kantons Zürich, welche von opendata.swiss heruntergeladen wurden. Als ZIP-Datei können die Shapefiles direkt in QGIS als Vektorlayer hinzugefügt werden. Zum Konvertieren müssen sie jedoch entpackt sein.

Es können stattdessen auch andere Daten verwendet werden. Hier gibt es keine Einschränkungen, bis auf, dass es sich um Shapefiles handeln muss.

Das ZIP-Dateiformat ist ein Format für verlustfrei komprimierte Dateien, das einerseits den Platzbedarf reduziert und andererseits mehrere zusammengehörige Dateien zusammengefassen kann.

Viele Befehle erwarten eine Eingabe (Input) oder eine Quelle und einige auch ein Ziel (oder Senke) oder eine spezielle Ausgabe (Output). ogrinfo verlangt eine Datei (ggf. mit Pfad) als Input (siehe oben). ogr2ogr erwartet einerseits den Namen des Inputs (Quellpfad) und dann den Namen des Outputs (Zielpfad) der zu verarbeitenden Dateien.

Übung 2: Shapefiles nach GeoPackage mittels ili2gpkg konvertieren

Zuerst wirst du die Interlis-Datei einlesen und somit das Schema in die Datenbank importieren.

Übung 2.1 Shapefiles nach GeoPackage konvertieren (Windows)

1. Öffne die OSGeo4W Shell.
2. Gib den Befehl `ogr2ogr` *outputfile inputfile* ein. *outputfile* tauscht du mit dem Zielpfad aus. *inputfile* tauschst du mit dem Quellpfad aus, wie z.B.: `"C:\Users\User\Downloads\kantonsstrassen_zh\kantonsstrassen_zh.shp"` Der Befehl könnte danach etwa so aussehen:
   `ogr2ogr "C:\Users\User\Downloads\kantonsstrassen_zh\kantonsstrassen_zh.gpkg" "C:\Users\User\Downloads\kantonsstrassen_zh\kantonsstrassen_zh.shp"`.
3. Nun sollte sich am angegebenen Zielpfad ein GeoPackage befinden, welches, wenn du es in QGIS öffnest, alle Kantonsstrassen aufzeigt.

Übung 2.1 Shapefiles nach GeoPackage konvertieren (macOS, Ubuntu)

1. Öffne dein Terminal.
2. Gib den Befehl `ogr2ogr` *outputfile inputfile* ein. *outputfile* tauscht du mit dem Zielpfad aus. Der Zielpfad, ist der Ort und Name des Endresultates, wie z.B.: `"~/Desktop/kantonsstrassen_zh/kantonsstrassen_zh.gpkg"` *inputfile* tauschst du mit dem Pfad der .shp-Datei aus, wie z.B.: `"~/Desktop/kantonsstrassen_zh/kantonsstrassen_zh.shp"` Der Befehl könnte danach etwa so aussehen:
   `ogr2ogr "~/Desktop/kantonsstrassen_zh/kantonsstrassen_zh.gpkg" "~/Desktop/kantonsstrassen_zh/kantonsstrassen_zh.shp"`.
3. Nun sollte sich am angegebenen Zielpfad ein GeoPackage befinden, welches, wenn du es in QGIS öffnest, alle Kantonsstrassen aufzeigt.

Damit kratzten wir aber nur an der Oberfläche der Möglichkeiten der OGR-Programmfamilie. Weitere Informationen findest du hier in der GDAL-Dokumentation oder der ogr2ogr-Dokumentation.

Kapitel 3: System-Prompt und Umgebungsvariablen

In diesem Kapitel lernst du, wie du den System-Prompt ändern kannst, indem du mit Umgebungsvariablen arbeitest und wie du gegebenenfalls Umgebungsvariablen festlegst.

In jeder Shell gibt es eine Eingabeaufforderung - genannt der **Prompt** - die dem Benutzer anzeigt, dass er einen Befehl ausführen kann. Ein Prompt besteht aus der eigentlichen Eingabeaufforderung als Zeichen und allenfalls weiteren Informationen davor. Windows benutzt für die Eingabeaufforderung das Zeichen `>`, Ubuntu and macOS zeigen in ihrer Shell ein `$`. Für allgemeine Zwecke einigen wir uns in diesem Arbeitsblatt auf das Dollarzeichen `$` als Prompt-Zeichen.

Eine **Umgebungsvariable** ist eine Variable mit einem Wert, auf die Prozesse Zugriff haben. Man kann diese Variablen für alle Benutzer geltend machen oder nur für einen bestimmten Benutzer. Ausserdem können Umgebungsvariablen auch nur für eine Sitzung in einer Shell gesetzt oder geändert werden. Die Variablen haben diverse Funktionen - von "Was ist das Home-Verzeichnis?" bis zu "Welche Dateiendungen kann die Shell ausführen?".

Zudem gibt es noch **Shell-Variablen**. Diese sind nur für den Shell-Prozess gedacht und deswegen auch nur für diese gesetzt.

Damit man Programme in einem CLI aufrufen kann, ohne den absoluten Pfad angeben zu müssen, kann man das Unterverzeichnis der `PATH`-Variable hinzufügen.

Falls du alle Umgebungsvariablen einsehen willst, benutze den Befehl `set` für Windows und `printenv` für Ubuntu/macOS. Dabei wirst du aber nicht die `PS1` Variable sehen, da diese eine Shell-Variable, welche noch zuerst mit `export` gesetzt werden muss, bevor sie eine Umgebungsvariable ist. Mehr dazu in der folgenden Übung.

Übung 3: Den Prompt anpassen

In dieser Übung wirst du das Prompt-Zeichen und den Text davor bearbeiten: * Übung 3.1 Den Prompt anpassen * Übung 3.2 Die Umgebungsvariable permanent setzen

Übung 3.1 Den Prompt anpassen (Windows)

Unter Windows heisst die Variable stattdessen `PROMPT`. Der Standardwert ist sehr kurz und unkompliziert. Rufe ihn mit dem Befehl `set PROMPT` auf und du bekommst die Rückgabe `PROMPT=$P$G`.

- `$P` Zeigt die aktuelle Partition, sowie den Pfad an.
- `$G` Ist das Prompt-Zeichen `>`.

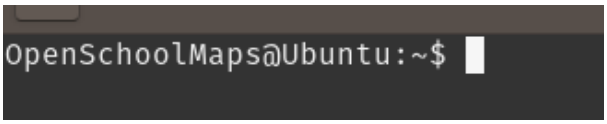
Die Variable kann einfach mit `PROMPT=...` gesetzt werden. Auch hier wird empfohlen auszuprobieren. Siehe diese Seite für eine Liste an Variablen, welche verschiedene Werte zurückgeben. Hier kann problemlos herumexperimentiert werden, da der Prompt rein kosmetisch ist und falls du deine Änderungen zurücksetzen willst, kannst du einfach das Terminal mit `exit` schliessen und wieder

öffnen.
Ansonsten kannst du den Beispielswert `$D$S$P$S$$` verwenden.

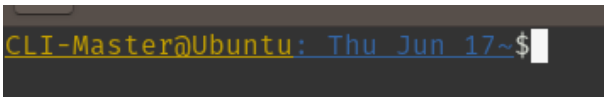
Auf Windows kann man leider nicht wie bei Bash (Ubuntu/macOS) individuell die Farben im Prompt anpassen, sondern nur Hintergrundsfarbe und Textfarbe. Verwende dafür den Befehl `color` *Farbe1Farbe2*. Um dein Terminal also dem Matrix-Theme entsprechend anzupassen, gib den Befehl `color 0A` ein. Benutze den Befehl `color --help` um eine Liste aller Möglichkeiten zu erhalten.

Übung 3.1 Den Prompt anpassen (macOS, Ubuntu)

Der Prompt wird über die Shell-Variable `PS1` definiert. Sie enthält einen Wert, welcher den Prompt so aussehen lässt:



Nachdem wir die Variable am Ende der Übung angepasst haben, könnte der Prompt danach so aussehen:



Schauen wir zuallererst einmal, was für einen Wert die Variable hat und analysieren diesen. Dies kannst du mit dem Befehl `echo $PS1` erreichen. Der Befehl `echo` zeigt einen Text an, das Dollarzeichen ist ein Prefix, der signalisiert, dass es sich beim folgenden Wert, um eine Variable handelt und, dass es dessen Wert nehmen soll.

Danach wirst du einen Output erhalten, der etwa so aussehen wird:
`\h:\W \u\$`

- `\h` Der Hostname (Name des Computers).
- `:` Doppelpunkt.
- `\W` Das aktuelle Verzeichnis, aber nur das hierarschisch gesehen unterste Verzeichnis.
- Danach folgt ein simples Leerzeichen.
- `\u` Der aktuelle Benutzer
- `\$` Zu guter Letzt das Prompt Zeichen. Da es einen Backslash vorne stehen hat, ändert sich der Prompt, wenn man als root angemeldet ist, von `$` zu `#`.

Und das ist die ganze Variable. Man kann sie anpassen wie man will, denn sie ist rein kosmetisch und hat keinen Einfluss auf das CLI an sich.

Die Variable (temporär) zu ändern ist sehr einfach. Der Befehl lautet: `export PS1="` ... `"` und mit ihm setzt du der Variable den angegebenen Wert. Dieser wird wieder zurückgesetzt, wenn du das Terminal wieder startest. Der Export setzt eine Variable als Umgebungsvariable mitsamt Wert.

Wir empfehlen dir selbstständig damit herumzuspielen. Auf dieser Seite findest du mehrere Variablen und Farbcodes, welche du ausprobieren kannst. Falls du deine Änderungen rückgängig

machen willst, kannst du mit dem Befehl `source ~/.bashrc` (Ubuntu) `/etc/bashrc` (macOs) zurücksetzen oder das Terminal neustarten. Der Befehl `source` führt das angegebene Skript in der Shell aus. `~/.bashrc` / `/etc/bashrc` ist der Pfad zu einer Bash-Konfigurationsdatei.

Ansonsten kannst du auch diese Beispielsvariable benutzen.
`\[\033[4;33m\]\u@\h\[\033[4;34m\]: \d\[\033[01;34m\]\w\[\033[00m\]\$`

Wie kurz angemerkt: Deine Änderungen sind nur temporär. Das liegt daran, dass jedes Mal wenn das Terminal gestartet wird, alle Variablen neu gesetzt werden.
Dies geschiet über die Konfigurationsdatei `.bashrc` / `bashrc`.

Übung 3.2 Die Umgebungsvariable permanent setzen

Wie mehrmals angesprochen, gehen die Änderungen jedes Mal beim Verlassen der Shell verloren und beim Starten werden die Vorgabewerte verwendet.

Übung 3.2 Die Umgebungsvariable permanent setzen (Windows)

Mit Windows ist das Setzen der Umgebungsvariablen sehr einfach. Nachdem man Mit dem Befehl `setx PROMPT Wert`, wobei `Wert` mit dem Variabelwert ersetzt wird.

Die Farbe kann leider nicht über CLI permanent verändert werden. Dafür kann man dies über die Benutzeroberfläche des cmd Programms erreichen, indem man auf den oberen Fensterrand rechtsklickt, dann **Kontextmenü › Eigenschaften** und dann dort die Farben anpasst.

Übung 3.2 Die Umgebungsvariable permanent setzen (macOS, Ubuntu)

Wie kurz angesprochen, werden die Variablen in Konfigurationsskripten für die Umgebung festgelegt. Diese Skripts werden jedes Mal beim Starten einies Terminals ausgeführt oder wenn man sie über den Befehl `source` ausführt. Man könnte eine seperate Skriptdatei schreiben, jedoch die einfachere, schönere Variante ist es, die Variable im `.bashrc`/`bashrc` zu definieren.

1. Die Datei bashrc (macOs) .bashrc (Ubuntu) sollte sich bei macOS im Verzeichnis `/etc` befinden und bei Ubuntu im Home-Verzeichnis `~`. Du kannst den Befehl `sudo find / bashrc` (macOS) `sudo find / .bashrc` (Ubuntu) verwenden, welcher im gesamten Dateiverzeichnis `/` nach der Datei `bashrc`/`.bashrc` sucht.
2. Nun öffne die Datei mit `nano /etc bashrc` (macOS) `nano ~/.bashrc` (Ubuntu).

```
if [ "$color_prompt" = yes ]; then
    PS1='${debian_chroot:+($debian_chroot)}\[\033[4;33m\]CLI-Master@\h\[\033[4;34m\]: \d\[\033[01;34m\]\w\[\033[00m\]\$ '
else
    PS1='${debian_chroot:+($debian_chroot)}\u@\h:\w\$ '
fi
unset color_prompt force_color_prompt

# If this is an xterm set the title to user@host:dir
case "$TERM" in
xterm*|rxvt*)
    PS1="\[\e]0;\u@\h: \w\a\]$PS1"
    ;;
*)
    ;;
esac
```

3. Wenn du eine neue Datei erstellt hast, kannst du einfach den Befehl `export PS1='Wert'`, wobei

`Wert` mit deinem gewünschten Wert ausgetauscht wird, einfügen. Ansonsten musst du nun in der Datei suchen, wo die PS1 Variable gesetzt wird. Dann musst du dort, wo die PS1 Variable zum ersten Mal gesetzt wird, den Wert mit deinem Wert austauschen.

4. Nun kannst du die Datei speichern `ctrl` + `O` und dann schliessen `ctrl` + `X`.
5. Jetzt kannst du die Shell neu konfigurieren mit dem Befehl `source ~/.bashrc` (Ubuntu) `/etc/bashrc` (macOS).

Du hast nun erfolgreich eine Umgebungsvariable gesetzt. So kannst du auch andere Umgebungsvariablen setzen; du musst sie einfach in einem Konfigurationsskript setzen, beziehungsweise direkt mit `setx` bei Windows.

Kapitel 4: Ein Java-Programm (ili2gpkg) als CLI ausführen

In dieser Übung wirst du eine weitere externe Java-Software über die Shell ausführen. Das Java-Programm heisst ili2gpkg und es macht eine Daten-Konversion von INTERLIS nach GeoPackage. Die Ziele dieser Übung sind, einerseits den Umgang mit absoluten Pfaden und den Umgang mit Umgebungsvariablen zu üben.

Pfade und Umgebungsvariablen

Alle Befehle (mit Ausnahme für die Shell essenzielle, wie z.B. `dir`) sind in einem Verzeichnis gespeichert. Bei Windows befinden sich die meisten in `C:\Windows\System32` als ausführbare Binärdateien (.exe) und bei macOS und Linux in `/bin`, `/sbin`, `/usr/bin` und `/usr/sbin` als Shell-Skripte (.sh).

Da wir ansonsten jedes Mal in der Befehlszeile also den ganzen Pfad angeben müssten, gibt es die **PATH**-Variable. Jedes Mal wenn ein Befehl ohne Pfad eingegeben wird, geht er die ganzen Verzeichnisse in der PATH-Variable durch. (Bei Windows wird sie auch für das "Ausführen"-Fenster verwendet (`Windowstaste` + `R`)

Das Problem ist, dass externe Software meistens im Betriebssystem spezifischem Applikationsverzeichnis gespeichert wird. Denn wenn die Verzeichnisse der PATH-Variable durchgegangen werden, werden nicht deren Unterverzeichnisse durchgegangen. Deswegen ist es von Vorteil ein Programm der Variable hinzuzufügen.

INTERLIS

INTERLIS ist ein in der Schweiz besonders in Geoinformationstechnologien verbreiteter Datenaustausch-Standard. Es ist ein systemneutraler Datenmodellierungs- und Transfermechanismus. Ein Interlis-Datenaustausch verlangt immer ein INTERLIS-Modell (.ili) oder mindestens einen Modell-Namen sowie das eigentliche Transferformat das bei der aktuellen INTERLIS Version 2 ein XML-Format ist. Mehr Informationen zu INTERLIS findest du auf interlis.ch.

Installation des Java Runtime Environments (JRE)

Für ein Java-Programm wie ili2gpkg wird eine Java Runtime Environment (JRE) benötigt. Diese JRE muss ggf. zuerst installiert werden und zwar mit der Version 1.6 oder neuer. Man kann auch das Java Development Kit (JDK) herunterladen, denn dies beinhaltet ein JRE.

Installation des JREs (Windows)

Prüfe zuerst, ob JRE nicht schon installiert ist. Gib dafür den Befehl `java -version` ein. Falls eine Fehlermeldung kommt in der Form "Der Befehl 'java' ist entweder falsch geschrieben oder konnte nicht gefunden werden.", dann ist Java nicht auf deinem Rechner installiert.

Lade also eine JRE hier herunter. Danach musst du noch den sogenannten 'bin' Ordner im Java-

Ordner der Path-Variable mitgeben, damit dein Terminal darauf zugreifen kann. Verwende dafür den Befehl `setx PATH "%PATH%;` gefolgt vom Pfad und noch ein Anführungszeichen `"`. Der Pfad könnte etwa so aussehen `C:\Program Files\Java\jdk-11.0.11\bin` und der Befehl `setx PATH "%PATH%;C:\Program Files\Java\jdk-11.0.11\bin`.

Danach musst du die JAVA_HOME-Variable erstellen und setzen. Programme, die mit Java arbeiten, verwenden sie, um Java zu orten. Setze sie mit dem Befehl `setx JAVA_HOME` gefolgt vom Pfad deines JDKs. Wenn du bei der Installation nichts verändert hast, sollte dieser so aussehen: `"C:\Program Files\Java\jdk-11.0.11\"`.

Teste jetzt wieder, ob Java installiert wurde, wie am Anfang der Übung.

Installation des JREs bzw. JDKs (macOS)

Prüfe zuerst, ob Java nicht schon installiert ist. Gib dafür den Befehl `java -version` ein. Falls eine Fehlermeldung kommt in der Form "Der Befehl 'java' ist entweder falsch geschrieben oder konnte nicht gefunden werden.", dann ist Java nicht auf deinem Rechner installiert. Wenn du die Meldung "No Java runtime present, requesting install." erhältst, klicke auf **[Weitere Infos ...]** und lade von dort eine JDK über **[JDK Download herunter]**. Sonst kannst du die JDK (enthält die JRE) hier herunterladen.

Führe die Installation durch. Merke dir den Speicherort des Java Ordners.

Nun musst du die Umgebungsvariablen PATH und JAVA_HOME mit der JDK erweitern. Die Path-Variable ist für das Terminal und Java_Home ist für andere Programme, die auf Java zugreifen.

1. Öffne die Konfigurationsdatei mit dem Befehl `sudo nano /etc/bashrc`.
2. Setze die Umgebungsvariable mit dem Befehl . `export PATH=$PATH:` gefolgt vom Pfad deines bin Ordners des JDKs. Der Pfad kann so aussehen `/usr/java/jdk1.6.0_10/bin`. Diesen Befehl fügst du am Ende der Datei ein.
3. Setze auch JAVA_HOME mit dem Befehl `export JAVA_HOME=` gefolgt vom JDK Ordner. Dieser kann so aussehen `/usr/java/jdk1.6.0_10`. Füge diesen Befehl direkt unter dem PATH export.
4. Speicher `ctrl` + `O` und schliesse `ctrl` + `X` die Datei.

Teste jetzt wieder, ob Java installiert wurde, wie am Anfang der Übung.

Installation des JREs bzw. JDKs (Ubuntu)

Prüfe zuerst, ob Java nicht schon installiert ist. Gib dafür den Befehl `java -version` ein. Falls eine Fehlermeldung kommt in der Form "Der Befehl 'java' ist entweder falsch geschrieben oder konnte nicht gefunden werden.", dann ist Java nicht auf deinem Rechner installiert.

1. Aktualisiere deinen Paket-Manager mit dem Befehl `sudo apt update`. Damit wird schergestellt, dass die neuesten Versionen der Pakete zur Verfügung stehen.
2. Installiere ein JDK mit dem Befehl `sudo apt install default-jdk`. Hier folgt der Installationsprozess, bei dem du noch mal deine Zustimmung geben musst.

Teste nun wieder, ob Java richtig installiert wurde. Nun musst du die Umgebungsvariable PATH mit

dem JDK erweitern.

1. Öffne die Konfigurationsdatei mit dem Befehl `sudo nano ~/.bashrc`.
2. Setze die Umgebungsvariable mit dem Befehl `export PATH=$PATH:` gefolgt vom Pfad deines bin Ordners von der JDK. Der Pfad kann so aussehen `/usr/java/jdk1.6.0_10/bin`. Diesen Befehl fügst du am Ende der Datei ein.
3. Setze auch JAVA_HOME mit dem Befehl `export JAVA_HOME=` gefolgt vom JDK Ordner. Dieser kann so aussehen `/usr/java/jdk1.6.0_10`. Füge diesen Befehl direkt unter dem PATH export.
4. Speicher `ctrl` + `O` und schliesse `ctrl` + `X` die Datei.

Teste jetzt wieder, ob Java installiert wurde, wie am Anfang der Übung.

Installation QGIS (Windows, macOS, Ubuntu)

Wie bereits im Kapitel 2 verwenden wir QGIS, um die konvertierte GeoPackage-Datei anschauen zu können.

⇒ Siehe das Arbeitsblatt von OpenSchoolMaps "Einführung in QGIS 3: Einleitung und Vorbereitung" für die Installation von QGIS.

Installation ili2gpkg (Windows, macOS, Ubuntu)

ili2gpkg ist ein Java-basiertes CLI-Werkzeug zum Konvertieren von INTERLIS-Daten zu GeoPackage und umgekehrt.

ili2gpkg ist hier herunterladbar.

Nachdem du ili2gpkg heruntergeladen hast, verschiebst du am besten den ganzen Ordner in dein Programmverzeichnis, Es Bedarf keiner weiteren Installation, als dass es an einem geeigneten Ort heruntergeladen und ausgepackt wird. Dann benennst du die .jar Datei im Ordner zu `ili2gpkg.jar` um und den Ordner von `ili2gpkg-4.4.2` zu `ili2gpkg`.

Unix-artige Systeme besitzen mehrere Programmverzeichnisse, für dieses Beispiel kannst du `/opt` benutzen für Ubuntu benutzen, da ili2gpkg ein Add-on ist und `/Applications` für macOS.

Übungsdaten

Die Übungsdaten sind auf dieser offiziellen Interlis-Seite zu finden.

Übung 4.1 PATH einrichten

Die PATH-Variable hat die Funktion, dass das Terminal einen eingegebenen Befehl in allen in der Variable stehenden Pfade sucht. Dies hat den Effekt, dass man nicht den ganzen Pfad eines Programms/Befehels eingeben muss, sondern man den Befehl/das Programm ohne Verzeichnisse eingeben kann. so wird aus `java -jar "C:\Program Files (x86)\ili2gpkg\ili2gpkg.jar"`, `java -jar ili2gpkg.jar` Mit Java Archives (JAR) funktioniert dieser Trick leider nur auf Windows, weswegen macOS und Ubuntu User jedes Mal den absoluten Pfad eingeben werden müssen und das Setzen

der Variable dort keinen Sinn macht.

Übung 4.1 PATH einrichten (Windows)

Diese Vor-Übung erleichtert das spätere Arbeiten mit dem Java-Programm.

1. Öffne als erstes die Windows-Suche (`Windowstaste` oder über das Startmenü).
2. Gebe `Umgebungsvariable` ein und klicke dann auf das Ergebnis "Systemumgebungsvariablen bearbeiten".
3. Du solltest dich nun im Reiter Erweitert in den Systemeigenschaften befinden. Klicke auf **[Umgebungsvariablen…]**.

Achtung, Systemvariablen können nur von Administratoren bearbeitet werden! Dies aus gutem Grund, denn es gibt kein "Undo". Falls du dir also nicht sicher bist, kannst du einfach deine Benutzervariablen ändern.

4. Wähle in der Variablenliste `Path` aus.
5. Klicke auf **[Bearbeiten…]**.
6. Nun sollte sich das "Umgebungsvariable bearbeiten" Fenster geöffnet haben. Klicke auf **[Neu]**.
7. Klick auf **[Durchsuchen…]** und suche dann nach dem ili2gpkg **Ordner** und klicke dann auf **[Ok]**.
8. Nun kannst die drei Fenster durch jeweils **[Ok]** schliessen.

Du hast nun erfolgreich die Path-Variable erweitert, wenn du jetzt dein Command Prompt **neu startest**, und dann `ili2gpkg.jar` eingibst, sollte das Programm starten.

Übung 4.2 INTERLIS nach GeoPackage konvertieren

Schritt 1

Zuerst brauchst du das Modell (.ili). Damit erzeugst du ein Datenschema in einer neuen GeoPackage-Datei. Das Datenschema bestimmt die Datenstruktur der darin verwalteten Daten. Wenn du die Umgebungsvariable auf Windows mit dem ili2gpkg-Pfad erweitert hast, kannst du anstatt dem ganzen Pfad, einfach `ili2gpkg.jar` einfügen. macOS- und Ubuntubenutzer sowie Windowsbenutzer, welche die PATH Umgebungsvariable nicht gesetzt haben, müssen bei jedem Befehl den absoluten Pfad der ili2gpkg.jar Datei angeben.

Gib den Befehl `$ java -jar /opt/ili2gpkg/ili2gpkg.jar --schemaimport --defaultSrsAuth EPSG --defaultSrsCode 2056 --dbfile RoadsSimple.gpkg RoadsSimple.ili` ein.

- `java -jar` ist der Befehl, um JAR-Dateien auszuführen.
- `/opt/ili2gpkg/ili2gpkg.jar` (Ubuntu) `/Applications/ili2gpkg/ili2gpkg.jar` (macOS) `ili2gpkg.jar` (Windows) ist die Datei, welche den Java Befehl ausführt.
    - `--schemaimport` ist einer Option, welche das Importieren eines Modells als Schema ermöglicht.

- `--deafultSrsAuth EPSG` setzt die SRS Authority für Geometriespalten zu EPSG.
- `--defaultSrsCode 2056` setzt den SRS Code für Geometriespalten, wo sich der Wert nicht pro Attribut ermitteln lässt.
- `--dbfile RoadsSimple.gpkg` ist die Option für die Datei, in welcher das Schema implementiert werden soll: die Datenbank. In diesem Fall handelt es sich hier um RoadsSimple.gpkg. Diese wird im aktuellen Verzeichnis erstellt, ausser es wird ein absoluter oder relativer Pfad angegeben. Relative Pfade benutzen den Ort der JAR-Datei als Startverzeichnis.
- `RoadsSimple.ili` ist das Modell. Auch hier muss der absolute/relative Pfad angegeben werden. Relative Pfade benutzen den Ort von der JAR-Datei als Startverzeichnis.

Dass der Prozess erfolgreich beendet wurde, erkennst du an der Nachricht "Info: ...done". Danach sollte am angegeben Verzeichnis die Datei `RoadsSimple.gpkg` erstellt worden sein.

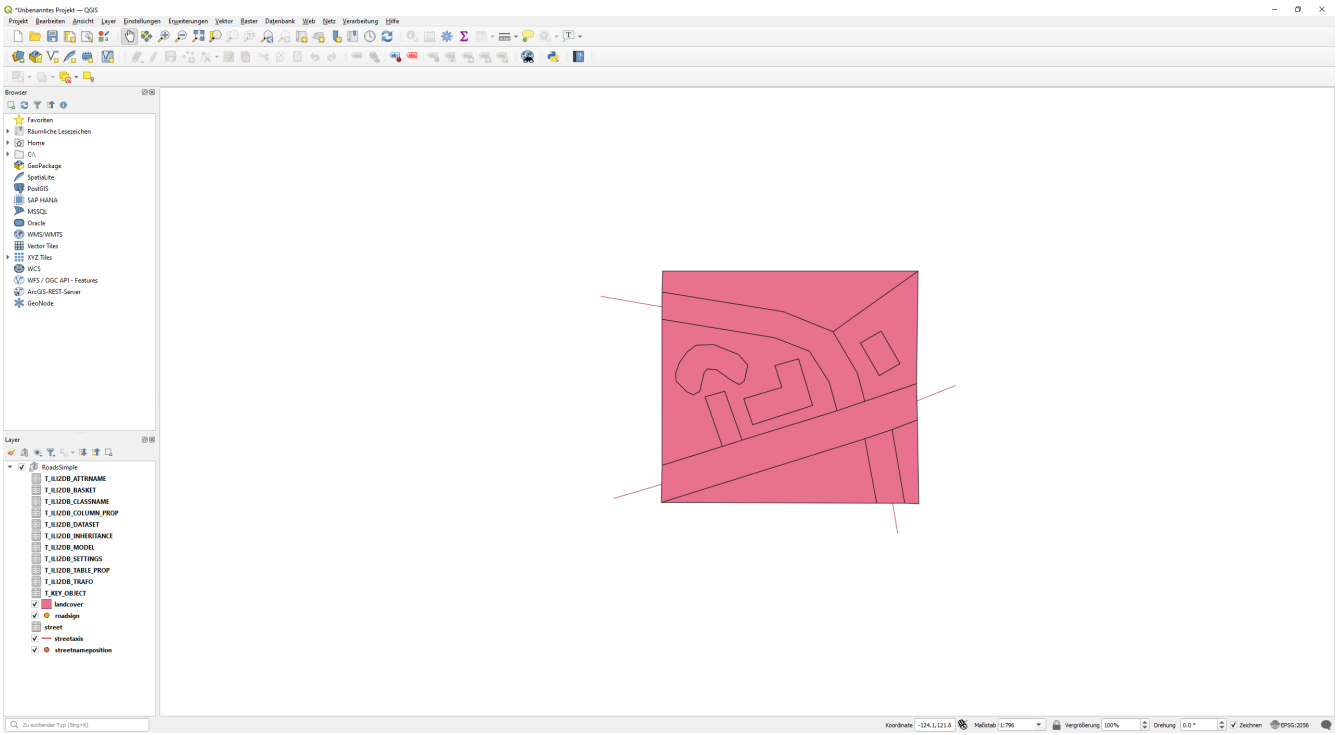
Schritt 2

Wenn du das GeoPackage jetzt in QGIS laden würdes, gäbe es noch nichts zu sehen, da erst die Struktur festgelegt ist; es fehlen noch die Daten.

Diese Daten kannst du mit dem Befehl `$ java -jar /opt/ili2gpkg/ili2gpkg.jar --import --dbfile RoadsSimple.gpkg RoadsSimple.xml` hinzufügen.

- `--import` weist das Programm darauf, dass nun Daten importiert werden sollen.
- `--dbfile RoadsSimple.gpkg` ist die Datenbank, welche du mit Daten füllen wirst. Achte darauf, dass du auf die selbe Datei verweist, wie beim Schemaimport.
- `RoadsSimple.xml` ist die Datentransportdatei, auch hier wieder auf den Pfad achten.

Dass der Prozess erfolgreich beendet wurde, erkennst du an der Nachricht "Info: ...import done". Danach kannst du das GeoPackage in QGIS öffnen.



Du hast nun erfolgreich INTERLIS-Daten in eine GeoPackage-Datei konvertiert. Damit kennst du einige Grundlagen vom Arbeiten mit CLIs und Umgebungsvariabeln, sowie auch, wie du externe Programme verwendest.

Noch Fragen? Siehe "Kontakt" auf OpenSchoolMaps!