

Geoinformatica

Developer's Guide

Ari Jolma

Geoinformatica Developer's Guide

Edition 24.10.2010

Author

Ari Jolma

ari.jolma@tkk.fi

Copyright © 2010 Ari Jolma This material may only be distributed subject to the terms and conditions set forth in the GNU Free Documentation License (GFDL), V1.2 or later (the latest version is presently available at <http://www.gnu.org/licenses/fdl.txt>).

Geoinformatica is a software stack for working with geospatial data. This is a developer's guide for writing geospatial programs to be executed from command line, in web servers, or as programs with graphical user interface. The guide covers the whole stack but focuses on the upper layers, which use Perl.

Preface	v
1. Document Conventions	v
1.1. Typographic Conventions	v
1.2. Pull-quote Conventions	vi
1.3. Notes and Warnings	vii
2. We Need Feedback!	viii
1. The Geoinformatica Software Stack	1
2. Perl	3
2.1. The Perl programming language	3
2.2. Some basic Perl modules used in Geoinformatica	3
2.3. The Gtk2 modules and namespace	3
2.4. The Geo::GDAL modules	3
2.5. Specific Geoinformatica Perl modules	3
2.6. Other Perl modules for geospatial	3
3. Cairo, GTK+ and GNOME	5
3.1. Cairo	5
3.2. GTK+	5
3.3. GNOME	5
4. GDAL	7
4.1. Proj4	7
4.2. GEOS	7
4.3. GDAL portability library	7
4.4. GDAL geospatial data model	7
4.5. Geospatial algorithms in GDAL	7
4.6. GDAL Perl bindings	7
5. libral	9
5.1. libral rasters	9
5.2. Raster algorithms in libral	9
5.3. Rendering of geospatial data in libral	9
6. Geo::OGC::Geometry	11
6.1. Test Section 1	11
7. Gtk2::Ex::Geo modules	13
7.1. Test Section 1	13
Index	15

Preface

1. Document Conventions

This manual uses several conventions to highlight certain words and phrases and draw attention to specific pieces of information.

In PDF and paper editions, this manual uses typefaces drawn from the *Liberation Fonts*¹ set. The Liberation Fonts set is also used in HTML editions if the set is installed on your system. If not, alternative but equivalent typefaces are displayed. Note: Red Hat Enterprise Linux 5 and later includes the Liberation Fonts set by default.

1.1. Typographic Conventions

Four typographic conventions are used to call attention to specific words and phrases. These conventions, and the circumstances they apply to, are as follows.

Mono-spaced Bold

Used to highlight system input, including shell commands, file names and paths. Also used to highlight keycaps and key combinations. For example:

To see the contents of the file **my_next_bestselling_novel** in your current working directory, enter the **cat my_next_bestselling_novel** command at the shell prompt and press **Enter** to execute the command.

The above includes a file name, a shell command and a keycap, all presented in mono-spaced bold and all distinguishable thanks to context.

Key combinations can be distinguished from keycaps by the hyphen connecting each part of a key combination. For example:

Press **Enter** to execute the command.

Press **Ctrl+Alt+F1** to switch to the first virtual terminal. Press **Ctrl+Alt+F7** to return to your X-Windows session.

The first paragraph highlights the particular keycap to press. The second highlights two key combinations (each a set of three keycaps with each set pressed simultaneously).

If source code is discussed, class names, methods, functions, variable names and returned values mentioned within a paragraph will be presented as above, in **mono-spaced bold**. For example:

File-related classes include **filesystem** for file systems, **file** for files, and **dir** for directories. Each class has its own associated set of permissions.

Proportional Bold

This denotes words or phrases encountered on a system, including application names; dialog box text; labeled buttons; check-box and radio button labels; menu titles and sub-menu titles. For example:

¹ <https://fedorahosted.org/liberation-fonts/>

Choose **System > Preferences > Mouse** from the main menu bar to launch **Mouse Preferences**. In the **Buttons** tab, click the **Left-handed mouse** check box and click **Close** to switch the primary mouse button from the left to the right (making the mouse suitable for use in the left hand).

To insert a special character into a **gedit** file, choose **Applications > Accessories > Character Map** from the main menu bar. Next, choose **Search > Find...** from the **Character Map** menu bar, type the name of the character in the **Search** field and click **Next**. The character you sought will be highlighted in the **Character Table**. Double-click this highlighted character to place it in the **Text to copy** field and then click the **Copy** button. Now switch back to your document and choose **Edit > Paste** from the **gedit** menu bar.

The above text includes application names; system-wide menu names and items; application-specific menu names; and buttons and text found within a GUI interface, all presented in proportional bold and all distinguishable by context.

Note the **>** shorthand used to indicate traversal through a menu and its sub-menus. This avoids difficult-to-follow phrasing such as 'Select **Mouse** from the **Preferences** sub-menu in the **System** menu of the main menu bar'.

Mono-spaced Bold Italic or ***Proportional Bold Italic***

Whether mono-spaced bold or proportional bold, the addition of italics indicates replaceable or variable text. Italics denotes text you do not input literally or displayed text that changes depending on circumstance. For example:

To connect to a remote machine using ssh, type **ssh *username@domain.name*** at a shell prompt. If the remote machine is **example.com** and your username on that machine is john, type **ssh *john@example.com***.

The **mount -o remount *file-system*** command remounts the named file system. For example, to remount the **/home** file system, the command is **mount -o remount */home***.

To see the version of a currently installed package, use the **rpm -q *package*** command. It will return a result as follows: ***package-version-release***.

Note the words in bold italics above — *username*, *domain.name*, *file-system*, *package*, *version* and *release*. Each word is a placeholder, either for text you enter when issuing a command or for text displayed by the system.

Aside from standard usage for presenting the title of a work, italics denotes the first use of a new and important term. For example:

When the Apache HTTP Server accepts requests, it dispatches child processes or threads to handle them. This group of child processes or threads is known as a *server-pool*. Under Apache HTTP Server 2.0, the responsibility for creating and maintaining these server-pools has been abstracted to a group of modules called *Multi-Processing Modules (MPMs)*. Unlike other modules, only one module from the MPM group can be loaded by the Apache HTTP Server.

1.2. Pull-quote Conventions

Terminal output and source code listings are set off visually from the surrounding text.

Output sent to a terminal is set in mono-spaced roman and presented thus:

```
books      Desktop  documentation  drafts  mss    photos  stuff  svn
books_tests Desktop1  downloads      images  notes  scripts svgs
```

Source-code listings are also set in mono-spaced roman but add syntax highlighting as follows:

```
package org.jboss.book.jca.ex1;

import javax.naming.InitialContext;

public class ExClient
{
    public static void main(String args[])
        throws Exception
    {
        InitialContext iniCtx = new InitialContext();
        Object          ref    = iniCtx.lookup("EchoBean");
        EchoHome        home   = (EchoHome) ref;
        Echo             echo   = home.create();

        System.out.println("Created Echo");

        System.out.println("Echo.echo('Hello') = " + echo.echo("Hello"));
    }
}
```

1.3. Notes and Warnings

Finally, we use three visual styles to draw attention to information that might otherwise be overlooked.



Note

Notes are tips, shortcuts or alternative approaches to the task at hand. Ignoring a note should have no negative consequences, but you might miss out on a trick that makes your life easier.



Important

Important boxes detail things that are easily missed: configuration changes that only apply to the current session, or services that need restarting before an update will apply. Ignoring a box labeled 'Important' won't cause data loss but may cause irritation and frustration.



Warning

Warnings should not be ignored. Ignoring warnings will most likely cause data loss.

2. We Need Feedback!

You should over ride this by creating your own local Feedback.xml file.

The Geoinformatica Software Stack

Geoinformatica is an interoperable stack of software. At the lowest layer of the stack there are libraries and parts of libraries that enable the developer to find information about and use resources of the current computer system and the network it is connected to. On the second layer there are several tools for various tasks like creating windows on the user interface, performing cartographic projection and topological computations, storing and managing data in long-term storage, and interpreting and compiling high-level program code. On the third layer there are several tools that provide increasingly generic and high-level interfaces to geospatial data and methods.

In the following chapters we will introduce and describe the central tools in the stack. The first two tools (Perl and GTK+) are very broadly used and not specific to geospatial. The third tool (GDAL) is a central piece of software for a large subset of free geospatial tools. The tools following the first three ones are specific to Geoinformatica.

Geoinformatica exploits heavily the idea of dividing computational tasks into more-simple-but-fast and complex-but-more-manageable-with-high-level-languages. The lower layers of Geoinformatica are mostly written in C or C++, while the upper parts are written in Perl. The access to libraries written in C (or other languages) from Perl is achieved by writing foreign function interfaces (FFIs). In some cases language-independent FFI tools, e.g., Swig, are used.

The repository and wiki for Geoinformatica-specific software is at <http://trac.osgeo.org/geoinformatica/>

Perl

Perl is a high-level programming language. Perl programs are written into text files, which the Perl executable interprets and compiles into intermediate language, which is then executed by a virtual machine. Perl programs are typically highly modular and each module may load dynamically binary libraries originally written in other languages like C. Perl modules are usually shared by the Perl community through CPAN.

2.1. The Perl programming language

To be written

2.2. Some basic Perl modules used in Geoinformatica

To be written

2.3. The Gtk2 modules and namespace

To be written

2.4. The Geo::GDAL modules

To be written

2.5. Specific Geoinformatica Perl modules

To be written

2.6. Other Perl modules for geospatial

To be written

Cairo, GTK+ and GNOME

Geoinformatica uses a mixed Cairo libral environment for rendering geospatial data. The goal is to increase the use of Cairo and eventually to move into a pure Cairo rendering. GTK+ and its components (GLib, Pango, GDK, etc.) are used for managing the rendered map and for creating graphical user interfaces (GUIs). Some essential code for interacting with Cairo and GDK is written in C, but everything else regarding GUIs the GTK+ Perl FFI (Glib, Pango, Gtk2, etc.) is used.

3.1. Cairo

To be written

3.2. GTK+

To be written

3.3. GNOME

To be written

GDAL

GDAL is a large piece of software for working with geospatial data. GDAL consists of several parts: a portability library, a generic system of classes for raster and vector data, a mechanism of adding format drivers and an increasing set of drivers, an increasing set of implementations of geospatial algorithms, several command line tools, and a foreign function interface (FFI) with bindings for several languages. Geoinformatica links to GDAL through the GDAL Perl bindings, i.e., the Geo::GDAL set of modules.

Besides several libraries to which GDAL links because they are required for drivers, GDAL also links to two important geospatial libraries: Proj4 and GEOS.

The vector data part of GDAL is called OGR. Thus GDAL is sometimes referred to also as GDAL/OGR.

4.1. Proj4

To be written

4.2. GEOS

To be written

4.3. GDAL portability library

To be written

4.4. GDAL geospatial data model

To be written

4.5. Geospatial algorithms in GDAL

To be written

4.6. GDAL Perl bindings

To be written

libral

The libral raster library serves three purposes in Geoinformatica. First, it is a simple and fast library for in-memory raster math. Second, it is a platform for implementing some geospatial raster algorithms. Third, it contains code for rendering geospatial data (libral-rasters and OGR vector data) on pixel buffers.

5.1. libral rasters

libral raster is a two-dimensional table of cells. A cell is a square area in map coordinates, which contains either a RAL_INTEGER or a RAL_REAL.

5.2. Raster algorithms in libral

To be written

5.3. Rendering of geospatial data in libral

To be written

Geo::OGC::Geometry

The Geo::OGC::Geometry implements the OGC simple features data model in Perl. It does not implement the computational geometry methods of the simple features standard.

6.1. Test Section 1

This is a test paragraph in a section

Gtk2::Ex::Geo modules

The Gtk2::Ex::Geo namespace contains 10 classes: DialogMaster, whose subclasses are Dialogs, Glue, History, Layer, Schema, Overlay, PseudoOverlay, Canvas, TreeDumper

7.1. Test Section 1

This is a test paragraph in a section

Index

F

feedback

contact information for this manual, viii

