

# **IDL Dynamically Loadable Module for Reading and Writing Analyze™ 7.5 Files**

Stephan Witoszynskyj\*

Release rel-20061124, Revision 1.4

\*E-mail: [stephan@wito.org](mailto:stephan@wito.org)

URL: <http://stephan.wito.org/>

This document is the documentation for a Dynamically Loadable Module (DLM) for IDL that provides functions to read and write data stored in the Analyze<sup>TM</sup> 7.5 Format.

This program is free software; you can redistribute it and/or modify it under the terms of the GNU General Public License as published by the Free Software Foundation; either version 2, or (at your option) any later version.

This program is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU General Public License for more details.

You should have received a copy of the GNU General Public License along with this program; if not, write to the Free Software Foundation, Inc., 59 Temple Place - Suite 330, Boston, MA 02111-1307, USA.

# Contents

- 1 Introduction** **4**
  
- 2 Installation** **5**
  - 2.1 Compiling . . . . . 5
  - 2.2 Installing the DLM . . . . . 5
  - 2.3 32Bit vs. 64Bit . . . . . 6
  
- 3 User Guide** **7**
  - read\_analyze . . . . . 7
  - write\_analyze . . . . . 7

# 1 Introduction

The IDL Dynamically Loadable Module described in this document provides functions to read and write Analyze™ 7.5 data to IDL. It is based on the Analyze™ 7.5 documentation [1]. Unfortunately, this documentation is not very clear about various details of the header and how certain fields in the header are to be treated. Therefore, the DLM just provides a very basic interface to the Analyze™ 7.5 format. Most of the fields in the header are being ignored. The functionality that is implemented is the same as in the example programs of the Analyze™ 7.5 documentation.

# 2 Installation

## 2.1 Compiling

Compiling the source code is straightforward. The only prerequisite is the `idl_export.h` file. Since, its location depends on the IDL installation the file `Makefile` in the source code's directory might have to be adjusted. By default the `Makefile` assumes that the environment variable `$IDL_DIR` is set and that `idl_export.h` is in the directory `$IDL_DIR/external/include`. If this is not the case the first line of `Makefile` has to be modified.

To compile the DLM type `make` (this, of course, has to be done in the source code's directory).

## 2.2 Installing the DLM

Installing the DLM is very simple. The files `analyze.dlm` and `analyze.so` have to be copied into a directory that is located within IDL's DLM search path. The search path is defined by the environment variable `$IDL_DLM_PATH` (the contents of this variable can be accessed by the IDL variable `!DLM_PATH`). By default this variable is not set and IDL uses its own directory as default value.

If an additional directory shall be added to the DLM search path, the contents of the environment variable `$IDL_DLM_PATH` has to be modified before IDL is started (an example is given in listing 2.1). Special care has to be taken, if this variable was not set before. In this case one also has to add IDL's default DLM search path explicitly to `$IDL_DLM_PATH`, because any path defined in the `$IDL_DLM_PATH` variable overrides IDL's defaults.

```
1| export IDL_DLM_PATH=$IDL_DLM_PATH:/opt/foo
```

Listing 2.1: Setting the IDL's DLM search path to the directory `/opt/foo` at the BASH command line.

## 2.3 32Bit vs. 64Bit

Special care has to be taken if the DLM is to be installed centrally for both 32-bit and 64-bit machines<sup>1</sup>. On a 32-bit machine the DLM has to be a 32-bit library and in case of a 64-bit machine the DLM has to be a 64-bit library<sup>2</sup>. Unfortunately, IDL expects the library file to be called `analyze.so` in both cases! The only solution is to install the 32-bit DLM and the 64-bit DLM in two different directories and adjust `$IDL_DLM_PATH` correspondingly.

---

<sup>1</sup>In this context “64-bit machine” means that a 64-bit operating system is installed. If this is not the case no precautions have to be taken.

<sup>2</sup>The easiest way of obtaining those two different DLMs is to compile the DLM once on a 32-bit machine and once on a 64-bit machine.

## 3 User Guide

### read\_analyze

Reads an Analyze data-set from disk and returns an IDL array of the same dimensions and data-type as the the data in the Analyze file. The Analyze data can be of following data types: 16-bit signed integer, 32-bit signed integer, float, double and complex (consisting of two float values).

#### Syntax

```
result = read_analyze(Filename [,VOXEL_SIZE=vector] )
```

#### Return Value

Result is an array having the same dimensions and data type as the data stored in the Analyze data-set. There is an exception to this rule: in case the size of the last dimensions is 1, IDL truncates those dimensions.

#### Arguments

**Filename** a string containing the name of the Analyze data-set which is to be read. The file-name can either be the name of the header file (having the suffix `.hdr`) or the image file (having the suffix `.img`) or the name without any suffix. In the first two cases the name of the missing file is obtained by replacing the suffix with `.img` or `.hdr` respectively. In the last case the respective suffices are added to obtain the name of the header and the image file.

#### Keywords

**VOXEL\_SIZE** This keyword takes a variable into which the voxel geometry is stored. The variable does not have to be set to a value before it is passed to the function. After execution the variable will contain a float array with the voxel geometry.

### write\_analyze

Writes an IDL array to disk as Analyze data-set. The array can have up to seven dimensions and be of the following data types: INTEGER, LONG, FLOAT, DOUBLE or

COMPLEX. The Analyze header created by `write_analyze` only contains information on the image dimensions and data type.

## Syntax

```
write_analyze , Filename , Data [,VOXEL_SIZE=vector]
```

## Arguments

**Filename** a string containing the name of the Analyze data-set which is to be read. The file-name can either be the name of the header file (having the suffix `.hdr`) or the image file (having the suffix `.img`) or the name without any suffix. In the first two cases the name of the missing file is obtained by replacing the suffix with `.img` or `.hdr` respectively. In the last case the respective suffices are added to obtain the name of the header and the image file.

**Data** an array containing the data which are to be stored. The array can have up to seven dimensions and be of one of the following data types: INTEGER, LONG, FLOAT, DOUBLE, COMPLEX. If the array is of any other data type the function returns an error message and the data is not saved.

## Keywords

**VOXEL\_SIZE** a one-dimensional vector that contains the voxel geometry (in mm). The vector can contain as many elements as the image has dimensions, but has to have at least three elements.

# Bibliography

- [1] Biomedical Image Resource, Mayo Foundation. *Analyze<sup>TM</sup> 7.5 File Format*.  
<http://www.mayo.edu/bir/PDF/ANALYZE75.pdf>.