ADVENTURE_IO

Input / Output format and libraries for ADVENTURE modules

Version 1.2

User's Manual

February 17, 2006

ADVENTURE Project

Contents

1. Outline

ADVENTURE_IO is a common input/output library supporting the data format, which is used by all modules of the ADVENTURE System. It is designed to provide:

- common data format for all ADVENTURE modules;
- ability of large data transfer;
- ability to use the ADVENTURE System in the decentralized/parallel computer environments;
- high efficiency of data input and output.

ADVENTURE_IO treats the units of data in a special format called *Document*. One data file can contain a number of *Documents*. Each *Document* is composed from 3 informative elements.

1. *Document ID*, which can be generated by provided library functions, is used to make the determination of *Document* unique.
2. *Property* part contains information about data types and options that will be passed to the program modules. The information is stored as combinations of "*Key=Value*".
3. *Raw Data* part, where the information on various physical values and coordinate system is stored in a binary format.

Special library functions are prepared for access the above-mentioned components.

The following sequence of inner operations are adopted to read the data using ADVENTURE_IO library:

- open the data file;
- open the *Document*;
- read the *Document*'s contents;
- close the *Document*;
- close the data file.

For those users who are going to adopt the ADVENTURE_IO library to their own programs, more detailed information on the data input/output functions is provided in the file `AdvDocument.pdf` for reference. If you are going to use the ADVENTURE System modules without changing them, the information necessary to install the modules is given below.

2. Installation Procedure

Extract the program package (`tar` archive) and execute the following commands from the top directory `AdvIO-<Version name>`:

```
% ./configure
% make
% make install
```

The default target directory for installation is `$HOME/ADVENTURE`. It can be changed by adding the option `--prefix=<install_dir>` to the `./configure` command. For example, if the configuration is done by

```
% ./configure --prefix=/usr/local   ,
```

the execution of `make install` will result in installation of the files into the directory `/usr/local`.

The list of options available with the `./configure` command can be displayed by executing the following:

```
% ./configure --help   .
```

3. Library Linkage

If you consider to compile a program which uses ADVENTURE_IO, the path to the `include` directory should be specified as

```
-I<install_dir>/include   ;
```

and the path to the `lib` directory should be specified as:

```
-L<install_dir>/lib   .
```

The library's header files will be installed (by default) in the directory `<install_dir>/inclue/Adv` and the libraries will be installed in the directory `<install_dir>/lib`. If necessary, the references on library's header files in the program's texts should be made as

```
#include <Adv/AdvDocument.h>.
```

The options `-lAdvDocIO -lAdvFileIO -lAdvBase` can be used for linking. These options can be also obtained by using the script `advsys-config` located in the directory `<install_dir>/bin`.

The appropriate path to the `include` directory (`-I`) can be obtained by adding the option `--cflags` to the command `advsys-config`:

```
% advsys-config --cflags   .
```

The appropriate path to the `lib` directory (`-L` and `-l`) can be obtained by adding the option `--libs <category>` to the command `advsys-config`:

```
% advsys-config --libs <category>  .
```

Here, the `<category>` can be selected from three options: `base`, `fileio`, and `docio`.

The programs can be also compiled using the script `advsys-config`.
For example:

```
% cc prog.c `advsys-config --cflags --libs docio` ...
```