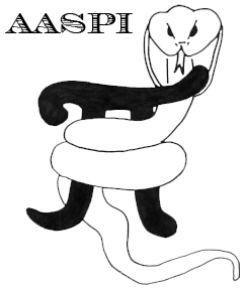


AASPI SOFTWARE STRUCTURE

Contents

Introduction

The AASPI software comprises a rich collection of seismic attribute generation, data conditioning, and multiattribute machine-learning analysis tools constructed by faculty and staff at the University of Oklahoma beginning in 2007. More recent additions have come from co-investigators at the University of Alabama, The University of Texas Permian Basin, and Sismo (http://mcee.ou.edu/aaspi/faculty.html). Because of the diversity of the interpretation software platforms used by our sponsors, the AASPI software is a stand-alone product and not intended to be tightly integrated with any specific interpretation package. Typically, the user exports SEGY-format seismic data volumes from their interpretation workstation, converts them to AASPI-format internal to the AASPI software package and computes a suite of attributes or attribute analyses. The resulting AASPI-format files are then converted back to SEGY-format files and loaded into the interpretation workstation. However, the software is modular enough that several sponsors have integrated the AASPI software into their own proprietary interpretation software packages. This topic will be discussed under the heading *Software Integration* below.

Sponsor Licensing, Software Modifications, and Ownership

Sponsors have access to all source code as well as compiled executables that run under Linux and Windows operating systems. Sponsors are free to use this software in making internal business decisions in their search for oil, or to provide service to commercial clients, partners, or host governments. The software license remains in perpetuity, with software updates and extensions being provided only to current sponsors. There is no limit to the number of

installations within the sponsors organization, or with their subsidiaries, as defined by the licensing agreement http://mcee.ou.edu/aaspi/Consortium_Agreement_Template/AASPI_Consortium_Agreement_Template_2018.pdf.

Sponsors are permitted to modify the code in any way, including incorporation in their internal non-commercial proprietary software packages. Software vendors are permitted to use the AASPI software as a template to help them improve or extend their own commercial software packages. In contrast, direct sale of the AASPI software requires a separate licensing agreement with the University of Oklahoma.

Software Location

The current version of the AASPI software will be under a website called mcee.ou.edu/aaspi/software. This site is password protected with a user name and password for each company. To avoid creation of confusion, this password is known to the person who is the company "champion" for the AASPI consortium and an identified IT contact who has root privileges on your network. People change assignments so please send an email at kmarfurt@ou.edu if you are the new person or need to know the password. We will normally send out a note to each sponsor (including the IT contact) when a new release has been made.

Software Structure

The AASPI software package consists of application programs, graphical user interfaces (GUIs), and scripts. Most of the application code is written in FORTRAN2005, all of the GUIs (including displays) are written in C++ using the Fox Toolkit (www.fox-toolkit.org), while all of the scripts are written in python. We use the Intel compiler to support the Message Passing Interface (MPI) parallelization libraries on both platforms. To avoid widespread dissemination of the source code, at present it can only be downloaded under the Linux subdirectory. The application codes, GUIs, and python scripts are identical for both Linux and Windows; only the method of compilation differs (e.g. using Makefiles under Linux and Visual Studio projects under Windows). Basically, the user invokes a GUI (typically from the master **aaspi_util** GUI) and selects appropriate files and defines appropriate parameters. This GUI then writes out a *.parms file that looks like a list of command line arguments. If successful, the GUI then invokes a python script. The python script reads the *.parms file and either parses them into command line arguments (for "interactive" or "fast batch") initiation, or into a batch program "job deck" appropriate for SLURM, PBS, or LSF batch job submission.

Programmers may initially think that the current separation of application programs, GUIs, and python scripts is cumbersome. However, there area several advantages to this structure:

1) Some programmers are proficient in Fortran, others in C++, and still others in python. By separating the codes, each part can be prototyped, tested, and debugged. For example, one can write an application and run it directly using command line arguments. Alternatively, one can write an application, a python script, hand edit a *.parms file, and run the program without having yet written a GUI.
2) Several sponsors have a Linux-based supercomputer linked to the same network as their Windows-based PCs. If the files are shared, the interpreter can invoke a Windows-based GUI on their desktop and when desired, submit a computationally intensive batch job (running SLURM, PBS, or LSF) on their supercomputer, and after completion, interactively plot the results back in Windows.

The software directory structure is almost the same for Linux and Windows except some specific external dependencies and build system.

Directory Name	**Contents**
bin64	Precompiled 64-bit AASPI executables
documentation	A local version of the documentation under http://geology.ou.edu/aaspi/documentation.html
ext_lib64	Linux Precompiled 64-bit support libraries (fftw. Fox Toolkit, intel64, openmpi)
ext_rpm	Source RPMs (Red Hat Package Manager) that may be needed
ext_src	Source archives and fixes for support libraries in Linux (fftw, Fox Toolkit, openmpi, SEPlib, SU).
ifort64	Window's Intel Fortran runtime libraries
mkl64	Windows' Intel Math Kernel Libraries runtime distribution, including FFTW and LAPACK
intelmpi	Windows' Intel MPI runtime libraries
include	Windows C++ include files (plus some AASPI 32-bit include files)
include64	AASPI 64-bit C++ include files
mod	AASPI 32-bit Fortran modules
mod64	AASPI 64-bit Fortran modules
lib64	AASPI precompiled 64-bit library files
lists	a suite of tables containing attribute names that controls the conversion of AASPI-format to SEGY-format files
maintenance	scripts and instructions to compile and release the software
par	AASPI default parameter files
pyscripts	AASPI Python (*.py) scripts
scripts	AASPI Bourne Shell (*.sh), SLURM, PBS, and LSF scripts.

sep_colors	Suite of color bars in SEP-format and *.alut format used by program AASPI plotting programs
src	The AASPI source code with a subdirectory for each application
visualstudioproject	Windows Visual Studio projects

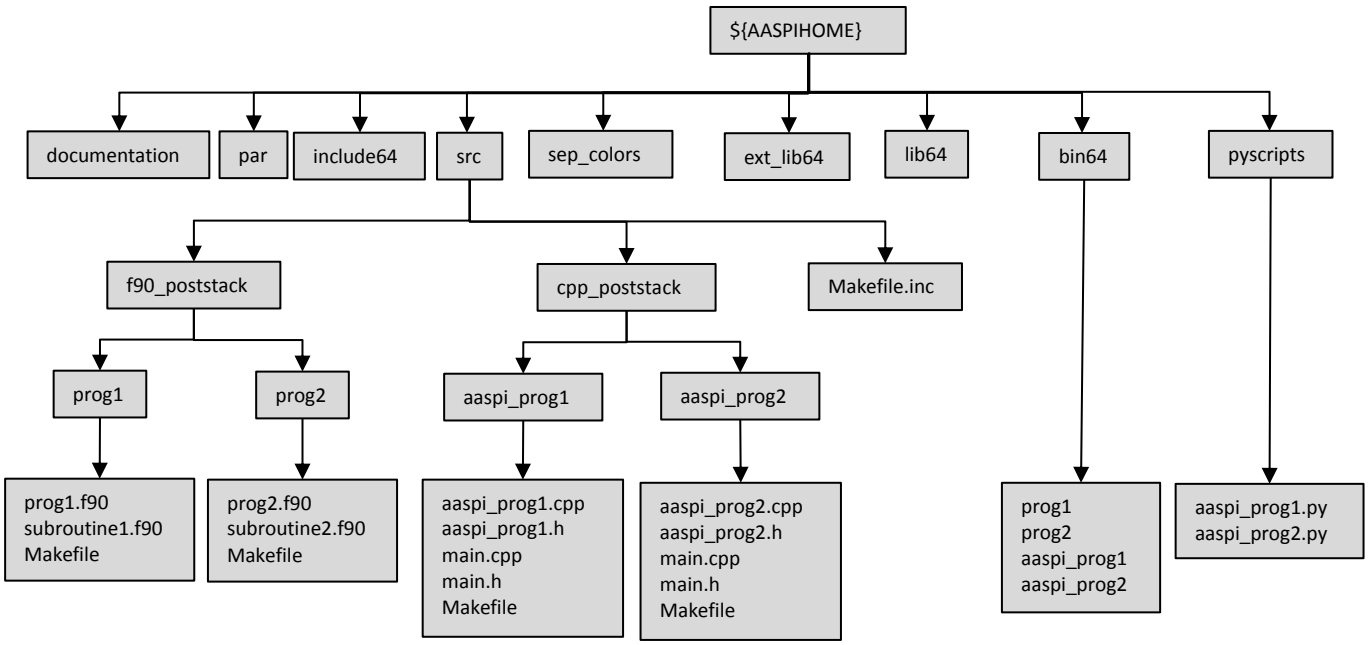
The *src* folder is identical in Windows and Linux and has the following structure:

Directory Name	**Contents**
cpp_lib	AASPI C++ library source codes (aapsi_io, aaspi_gui_lib)
cpp_poststack	AASPI C++ post-stack GUI source codes
cpp_prestack	AASPI C++ pre-stack GUI source codes
f90_lib	AASPI Fortran90 library source codes (lib_new_reorg, or fortran_mod in Windows)
f90_poststack	AASPI Fortran90 post-stack application source codes
f90_prestack	AASPI Fortran90 pre-stack application source codes
linux_only	Other AASPI GUI and application source codes that have not been ported to Windows

Several 3rd party binary libraries are used by the AASPI code: OpenMPI – Message Passing Interface, and FFT-W – the Fastest Fourier Transform in the West.) We do not directly use SEP – Stanford Exploration Project, or SU – Seismic Unix libraries, since neither of these, nor "Madagascar" will run under Windows. However, these software packages will read most AASPI-format files. The Fortran90 compiler support libraries *intel64* are also distributed with this distribution. Currently we are using the Intel Fortran Compiler version 12.1.2 for Linux and version 2015 update 1 for Windows. If you intend to use a different Fortran compiler, you will need to recompile all AASPI programs and provide the corresponding external dependencies (i.e. runtime distributions). We can provide compiled versions using former or future versions of Redhat on request.

The AASPI Software Directory Structure

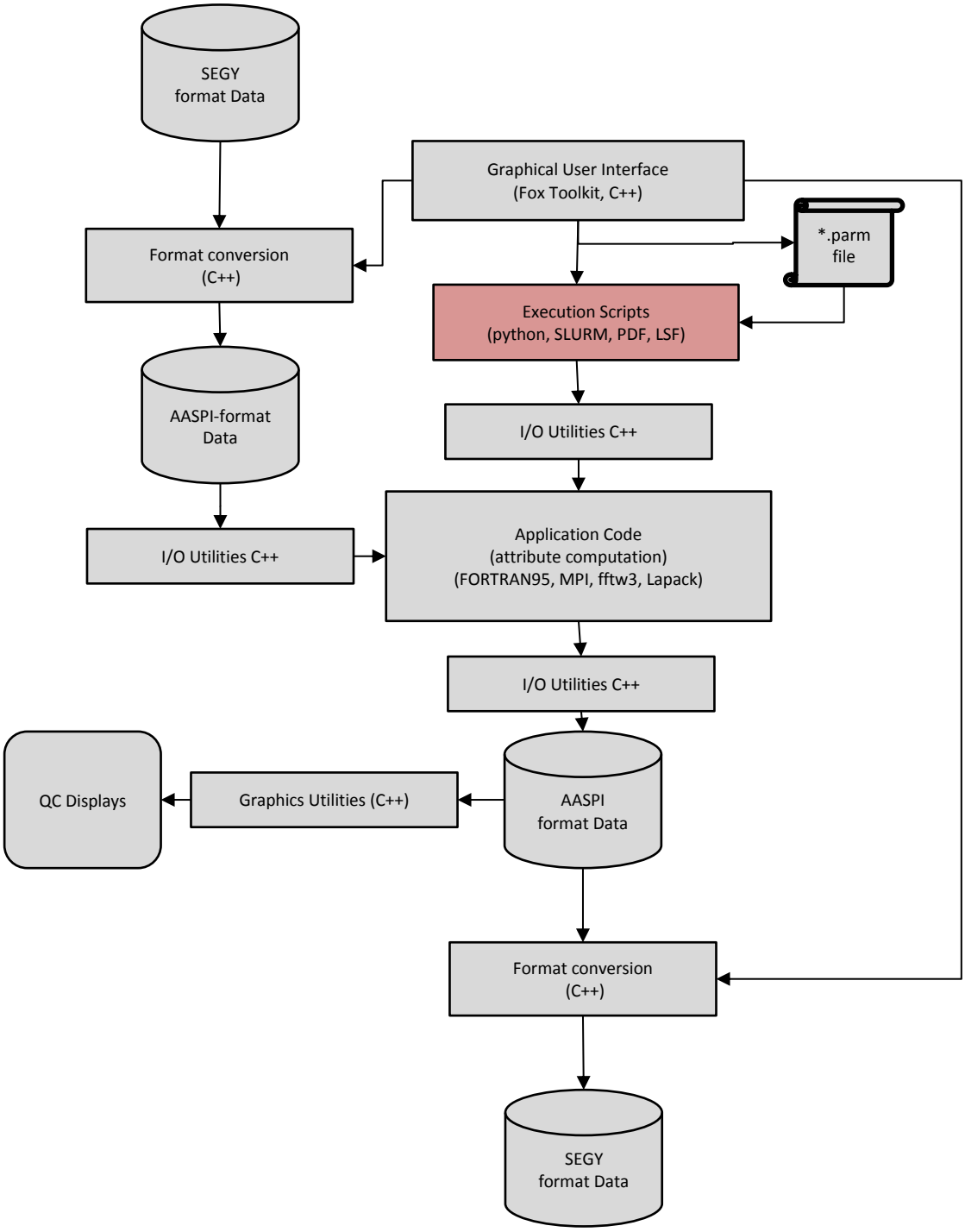
Although we try to be very explicit in our documentation about algorithm implementation and assumptions, algorithm developers and tech group staff may wish to examine details of how the software has been implemented. The figure below shows a subset of the directory structure described above (where the subdirectories f90_prestack and cpp_prestack are not displayed to make the figure simpler):



At the University of Oklahoma, we use public domain *git* to control our software development and deployment (http://git-scm.com) giving rise to the tree structure under src shown above. The Fortran90 source code is found under the f90_poststack and f90_prestack subdirectories. Within each of these are a suite of application programs (e.g. program *dip3d.f90*). These Fortran90 application programs are controlled by a graphical user interface or GUI which are found under the *cpp_poststack* and *cpp_prestack* directories. Our naming convention is simple, by adding the prefix *aaspi_* before each program name (e.g. *aaspi_dip3d.cpp*).
In Linux, each program has a simple Makefile. These Makefiles include a file called ../../Makefile.inc where all the installation specific paths and external libraries are defined. If you recompile the software you will probably need to modify the *src* level *Makefile.inc* file but *not* any of the program level Makefile files. (See optional compilation box below).
In Windows, each program has a corresponding Visual Studio project under *$AASPIHOME/visualstudioproject.*

Following common programming conventions, after compilation the binary codes are copied to the directory bin64 with names corresponding to the application (e.g. **dip3***d* and **aaspi_dip3d**) but *without* the extensions of *.f90* or *.cpp*. Intermediate object and executable code (not shown in the tree above) are stored under an obj64 directory under each application or GUI. The GUIs residing in the bin64 directory are typically invoked from master GUIs (such as *aaspi_util* and *aaspi_util_prestack* but can also be directly invoked by typing the binary file name (e.g. *aaspi_dip3d*). In most cases, each GUI will invoke a python script with a corresponding name (e.g. *aaspi_dip3d.py*) and output a parameter file in the user directory (e.g. *dip3d.parms*). The python script reads in the parameter file, parses the arguments into command line arguments and then executes the corresponding program (e.g. **dip3d**).

A Programmer's View of a Typical AASPI Flow



Recompiling the AASPI software (Optional)

90% of our current sponsors use the AASPI software "as is". Some "cherry pick" key ideas by inserting or reworking AASPI source code into their own application development environment. In either of these two situations you will *not* need to recompile the AASPI software.

Reasons for recompiling may include adding additional capabilities or improvements to the current version of the software, better linkage to your commercial interpretation package, or in at least one environment, recompiling using a machine-specific set of optimized libraries. In our environment, the caretakers of the OU supercomputer center (Oscer) have installed tuned versions of **OpenMPI, fftw3**, and **lapack**. Linking to these optimized libraries requires recompiling.

In order to compile you will need to determine the name of your compiler (in our case at OU the Intel compiler is called *ifort64* for 64-bit architecture. GNU Fortran compiler has limited support for Windows, so that's why we choose Intel Fortran compiler). You will also need to learn the absolute paths of each of the libraries that may be replaced.

For Linux:
The only file that needs to be modified is *${AASPIHOME}/src/Makefile.inc* (see tree structure above). You will note that the variable F90 is set to ifort64 at OU for 64-bit compilation. Change it appropriately. More troublesome is determining where your external libraries reside. They will need to be explicitly defined using the *complete path name*, not some local short cut. You may find that the environment variables in your *xterm* window are quite different that when the *Makefile* is invoked. We will print out these variables in the June 2014 release of our software to allow you to check these paths.

Once these variables have been set, change to the compilation directory by typing

cd ${AASPIHOME}/maintenance/deploy_scripts

Then type

python 01_make_aaspi_linux.py -c

The *01_make_aaspi_linux.py* script will compile programs defined by two files: *aaspi_gui_list* and *aaspi_application_list.* If you add new programs and wish to distribute to another site you may wish to augment these lists.

For Windows:
You just need to make sure you have the correct Visual Studio and Intel Fortran compiler. At OU, we use Visual Studio 2013 and Intel Fortran compiler 2015 update 1. If you use a different version of the Intel Fortran compiler, you also need to point to the equivalent runtime libraries for **ifort64**, **mkl64**, and **intelmpi**.
To recompile in Windows, simply run *aaspi.sln* under *$AASPIHOME/visualstudioproject*, set the *build mode* to *Release-x64*, and right click on *aaspi solution* in the project tree and choose *"Rebuild".*