MDSplus Quality Improvement Project

Thomas W. Fredian, Joshua A. Stillerman, Gabriele Manduchi\textsuperscript{1},
Andrea Rigoni\textsuperscript{1}, Keith Erickson\textsuperscript{2}

\textsuperscript{1}Consorzio RFX, Euratom-ENEA Association, Corso Stati Uniti 4, Padova 35127, Italy
\textsuperscript{2}Princeton Plasma Physics Laboratory, Princeton, NJ 08543, USA

June, 2015

Plasma Science and Fusion Center
Massachusetts Institute of Technology
Cambridge MA 02139 USA

This work was supported by the U.S. Department of Energy, Grant No. DE-SC0012470. Reproduction, translation, publication, use and disposal, in whole or in part, by or for the United States government is permitted.
MDSplus is a data acquisition and analysis system used worldwide predominantly in the fusion research community. Development began 25 years ago based on the OpenVMS operating system. Since that time there have been many new features added and the code has been ported to many different operating systems. There have been contributions to the MDSplus development from the fusion community in the way of feature suggestions, feature implementations, documentation and porting to different operating systems. The bulk of the development and support of MDSplus, however, has been provided by a relatively small core developer group of three or four members. Unfortunately, much more effort was focused on providing new features for the community than on keeping the underlying code and documentation up to date with the evolving software development standards. To ensure that MDSplus will continue to provide the needs of the community in the future, the MDSplus development team along with other members of the MDSplus user community has commenced on a major quality improvement project. The planned improvements include changes to software build scripts to better use GNU Autoconf and Automake tools, refactoring many of the source code modules using new language features available in modern compilers, using GNU MinGW64 to create MS Windows distributions, migrating to a more modern source code management system, improvement of source documentation as well as improvements to the www.mdsplus.org web site documentation and layout, and the addition of more comprehensive test suites to apply to MDSplus code builds prior to releasing installation kits to the community. This work should lead to a much more robust product and establish a framework to maintain stability as more enhancements and features are added. This paper will describe these efforts that are either in progress or planned for the near future.

Keywords: Data Acquisition Systems, Data Management, Data Formats, MDSplus

1. Introduction

MDSplus[1] is a collection of libraries and applications used for providing access and storage of scientific data. It provides a wide range of functionality including:

- Data acquisition from measurement devices; organization of the storage of both experimental measurements, analysis results and various metadata associated with those data items
- Access to the data and metadata from a wide variety of programming languages and utilities
- Remote data access using a variety of transport mechanisms.

Originally developed on the OpenVMS[2] platform in the 1980’s, MDSplus has since grown in functionality and has been ported to numerous computing platforms. MDSplus is used by scientists and engineers worldwide mostly in the field of fusion energy research.

The development and support of MDSplus has been accomplished by a relatively small core group and much effort was targeted on adding new functionality to the system requested by the user community and providing support for a growing number of computing platforms. More recently we have increased the number of core developers adding more expertise in modern coding standards and compiler capabilities and have shifted the focus to improve the quality of the core MDSplus product utilizing a variety of techniques discussed in this paper.

2. OpenVMS origins

As mentioned above, MDSplus was originally developed on the Digital Equipment Corporation’s OpenVMS operating system. This system provided a wide range of utilities which developers could build upon to provide specialized applications. MDSplus was designed and built to take advantage of these tools provided by the operating system. When it was later decided to port MDSplus to other operating systems it was anticipated that OpenVMS would be the predominant OS for MDSplus use. Instead of making major changes to the MDSplus code, the port to other operating systems was accomplished by emulating the utilities provided by OpenVMS. Surprisingly, at the time, OpenVMS use began to rapidly decrease and soon most sites migrated from OpenVMS to linux based operating systems. Today essentially all sites using MDSplus have moved off of OpenVMS. While MDSplus still functions and performs well on linux systems, its code base is still littered with OpenVMS
indicating potential problems or extraneous house addition, the code will be analyzed using the compiler's operating system features can be eliminated as well. In platforms so many of the configuration tests for are for the most part standardized across most of the easier to read and support. The GNU library functions the complicated conditional compilation sections of the standardizing on the GNU compiler, much if not all of by default or at least have GNU compilers available. By compiler being used. Today, all of the platforms that conditional compilation sections based on the particular idiosyncrasies. The MDSplus code became riddled with platform specific compilers each with their own systems in the late 1990's most other platforms had their

4. GNU compiler standardization

When MDSplus was ported to other operating systems in the late 1990’s most other platforms had their platform specific compilers each with their own idiosyncrasies. The MDSplus code became riddled with conditional compilation sections based on the particular compiler being used. Today, all of the platforms that MDSplus currently supports either use GNU compilers by default or at least have GNU compilers available. By standardizing on the GNU compiler, much if not all of the complicated conditional compilation sections of the MDSplus code can be removed making the code much easier to read and support. The GNU library functions are for the most part standardized across most of the platforms so many of the configuration tests for operating system features can be eliminated as well. In addition, the code will be analyzed using the compiler’s ability to display a wide variety of warning messages indicating potential problems or extraneous house cleaning problems such as old variables being declared but no longer being used. This compiler based cleanup of the code is another focus of the quality improvement project.

5. Standardized code indentation

During the ongoing development of MDSplus over the last 25 years there have been many different contributors to the code each with their own favorite coding styles. There was no dictated standard for indentation or commenting in the code. To improve the ability of all the developers to support all the code in MDSplus we are now using a common indentation style and using an utility called “indent”[7] which understands the coding language and can automatically indent the source code based on a selected indentation style. This has been used to indent all of the MDSplus code.

6. Standardized code documentation

Since the time when most of the MDSplus code was developed, new tools have been designed which can parse the code and find documentation fragments with specialized markup language that can be used to produce nicely formatted documentation, either web based or printed. We are in the process of adding better code documentation and standardizing on the use of the Doxygen[8] tool to parse annotated sources to produce source code documentation.

7. Code testing

During the current automated build and release process several basic regression tests are performed on the MDSplus code to ensure that the basic functionality is working. These tests are very limited in scope and only test a small subset of the utilities and function provide by MDSplus. Part of the improvement project is to expand the scope of these tests to provide improved test coverage. The tests will be incorporated as an option to the standard build procedures of MDSplus if appropriate. Using the gcov[9] tool (“--coverage” option ) of the GNU compiler it is possible to measure the percentage of the executable code is exercised by tests. Other test utilities such as valgrind[10] and the “-fsanitize” compiler option will be used to identify problems such as memory leaks and references to uninitialized memory which can produce intermittent problems. These tests should greatly improve the quality of the MDSplus releases and will ensure that major rewrites of existing code do not alter the existing behavior of the code.

8. Refactored code

Some of the major internal code of MDSplus could benefit by a complete rewrite of the code. One area of focus of the quality improvement project will be the reimplementation of the TDI expression evaluator. MDSplus was designed using a very object oriented approach and the expression evaluator is a good example of this. Unfortunately when it was developed most object oriented programming languages were in their infancy and tended to perform poorly. Today the c++ language is
widely used and is very suited for developing the TDI expression evaluator. It is crucial that a comprehensive test suite is developed for TDI to ensure that there is no change in execution behavior after it is replaced by a more modern implementation in c++.

9. Improved build tools

The configure and build process for MDSplus is also a focus of the quality improvement project. The hand coded construction of the configure and Makefile’s used in MDSplus will be replaced by the use of tools such as automake[11] and libtool[12] which provide standardized methods for describing how to build and install libraries and executables. The MDSplus automated build and release system has also been upgraded to use docker[13], a utility which enables you to construct pseudo virtual machines in which to build releases for different platforms. These docker virtual machines are much easier to construct and operate for this purpose than true virtual machine products.

10. Improved source code management

As part of the quality improvement project, the code management of MDSplus has been moved from cvs to git[14]. Git provides many features that makes it much easier to work in an environment with multiple developers and to handle release management. Much of the MDSplus release management scripting is much simpler after migrating to the use of git instead of cvs. Utilizing the central repository provided by github.com also simplifies the management of source code downloads.

11. Improved user documentation

A new MDSplus tutorial is being developed which will include working examples of applications and scripts which interact with MDSplus. These examples will be available as source downloads so users can build and run the examples on their local systems. As mentioned above new user documentation has been added to the MDSplus command utilities via a help command.

12. Conclusions

The MDSplus system has proved to be quite useful for the fusion community over the past quarter century and is used in a variety of ways in most fusion energy research sites worldwide. Beginning its existence as a package developed solely for the OpenVMS operating system it has been ported to many different computing platforms. It has grown considerably in functionality over the years and most of the focus of the developers was placed on adding new features and providing support for more and more platforms while fixing bugs in the code when they were detected. More recently this focus has changed toward improving the quality of the code which makes up MDSplus utilizing more modern coding standards and taking advantages of available software utilities which did not even exist in the early days of development of MDSplus. This change of direction occurred mostly because of very useful suggestions and comments from the MDSplus user community. We added two new core developers, Keith Erickson of PPPL and Andrea Rigoni of Consorzio RFX, who are assisting in this work and bring a lot of exciting new expertise which the original developers were lacking.

The use of more readily available standard coding utilities for software project building, coding and documentation will enable us to throw out large portions of the original MDSplus code making it a much more robust product while reducing the difficulty of maintenance. Using better documentation tools for embedding comments in the code will not only improve the experience for users of the product but will also ensure the development and maintenance of MDSplus can continue into the unforeseeable future.

Acknowledgments

The success of MDSplus stems largely from the suggestions and comments from the user community and this project is just another example of this collaboration of the developers with the users to provide a useful tool for the community.

A portion of this work is being performed under US Department of Energy Award Number: DE-SC0012470.

References