Requirement Specifications of an HDF5 File Format Validation Tool

Albert Cheng

1 Document purpose

This document describes the requirements of functionalities, implementation, tests and user documents of the project to implement an HDF5 file format validation tool, called h5chk. The details of design and implementation are covered in a different document. The following sections describe the requirements of each area of the project.

- Tool Requirements and Functional Specifications
- Implementation Requirements
- Tests Requirements
- Documents Requirements

2 Tool Requirements and Functional Specifications

2.1 Purpose of the h5chk tool

H5chk verifies an HDF5 file against the defined HDF5 File Format Specification [i](referred to as the *File Format* in this document). An HDF5 file is considered valid if it does not contradict the File Format; otherwise it is considered as invalid.

2.2 Why is it needed?

HDF5 is a platform independent data file format and can be used for short-term data files (e.g., application restart files) or long-term data management (e.g., NASA EOS data files). The HDF5 File Format Specification defines the data model and file format of an HDF5 file. The HDF5 library implements the Application Programming Interface (API) according to the File Format. It is important that HDF5 files created and modified by the HDF5 library are fully compliant with the File Format to ensure the data model integrity and long term compatibility between evolving versions of the HDF5 library.

The HDF Group is also submitting the HDF5 File Format Specification as an ANSI standard. The standardization process requires us to provide a verification tool that can confirm whether an HDF5 file conforms to the File Format standard.

The h5chk tool is envisioned to verify that the content of an HDF5 file is encoded according to the File Format. The verification role makes h5chk act as a watchdog for the implementation correctness of the HDF5 library. The most likely way to use the h5chk tool is to verify data files produced by a version of HDF5 library. A positive result would confirm both the correctness of the data files and also that particular version of library. Another important use is when a version of HDF5 library rejects a supposedly valid HDF5 data file, the h5chk can be used to verify the data file. A negative result would confirm the file is invalid and may even pinpoint the invalid parts. However, a

positive result would indicate that the data file is a valid HDF5 file but the particular version of HDF5 library contained implementation errors.

2.3 Functional Requirements

The h5chk tool must be able to read an HDF5 file, verify its content against a version of the File Format and report if the file is in full compliance with that version of the File Format. If the h5chk tool considers the file is not compliant with the File Format, it should report all non-compliance it detects.

2.4 Users Characteristics

The h5chk tool will be used by both HDF5 application users and HDF5 library developers. For example, application users may use h5chk to verify the data files are compliant to the File Format to ensure compatibility with different implementations of the HDF5 library; while library developers may use the tool to confirm the library is compliant to the File Format.

2.5 User Interface

2.5.1 Standalone tool

The h5chk tool will be a standalone tool with various command line options. It is invoked as a command and results are printed to the standard output in plain text. It should exit with the zero code if it detects no error, otherwise non-zero.

2.5.2 Library module

The internal code of h5chk should be organized as library modules with defined public API's. Other applications such as MATLIB may call the h5chk API's with proper parameters to do the validation.

2.6 What h5chk does not do

First of all, the h5chk tool does not verify if the File Format is correct, logical or complete. Though during the implementation of the tool, the programmer may detect deficiencies or even errors in the File Format, the h5chk tool is not meant to do this sort of analysis.

Secondly, the tool does not verify the correctness of file content that is not specified by the File Format. An obvious example is that h5chk does not verify the correctness of the raw data of a dataset. If an application stores the integer values with the wrong signs, for example, the h5chk tool will not detect that.

Lastly, the h5chk detects and reports errors. It does not correct errors it finds. This means h5chk only needs read permission to the data file.

3 Implementation Requirements

3.1 External Libraries and Algorithm Requirements

The h5chk tool code should be totally independent of the HDF5 library and does not use any HDF5 public or private API calls. It may not link with the HDF5 library or use its header files but it may use other external libraries (e.g., zlib) that the HDF5 library uses.

The h5chk tool code may adapt some of the algorithms or structures used by the HDF5 library if it is appropriate to do so.

3.2 Platforms Requirements

The h5chk tool should be available for all platforms where the HDF5 library is available. This requires that the implementation of the tool must be as platform independent as possible.

3.3 Programming Language Requirements

In order to ensure the h5chk tool is available on most platforms, a commonly available programming language should be used for the coding. Currently, the C programming language would be the appropriate choice.

3.4 Runtime Requirements

The h5chk tool should require only a moderate amount of memory to operate. For example, storing the entire file content in memory is not an acceptable implementation design. A rule of thumb is that the tool should require runtime memory no more than 20% of the size of the data files that are 1GB or larger.

The tool should require only a moderate amount of execution time to operate. A rule of thumb is that it should verify a data file in no more than twice the amount of time that the h5dump tool needs to process the same file.

3.5 Format Grammar driven

It is desirable to implement h5chk as Format Grammar driven, provided the File Format can be “coded” as some computer language grammar.

3.6 Target File Format Version

The tool should be implemented using version 1.1 of the HDF5 File Format.

4 Test Requirements

Tests should be coded to verify that the h5chk tool works correctly on both valid and invalid HDF5 data files.

4.1 Valid HDF5 Files

The h5chk tool should verify all valid HDF5 data files it examines with positive result. Valid HDF5 files would be created by some *file generators* or collected from real life HDF5 applications already in production.

4.1.1 Via File Generator

A set of HDF5 file generators would be written to create valid HDF5 files using various features of the HDF5 library and components of the File Format. These types of generated data files provide a systematic coverage of all components of the File Format. The h5chk tool must verify them to be in compliance to the File Format. Note that the file generators would be using the HDF5 public API's and linking with the HDF5 library.

4.1.2 Via Real Life HDF5 Applications

Production data files to be tested by the h5chk tool should be collected from different HDF5 applications inside and outside of the HDF Group. These types of application data files, though may possibly not covering all the components of the File Format, have the important advantage of being real life examples and may contain interesting uses of the components of the File Format that would be difficult to be produced by file generators.

4.1.3 Test against different versions of HDF5 library

Valid test data files should be created by multiple versions of HDF5 library. At least the latest official released version and a relatively stable development version of the library code should be included in this class of tests.

4.2 Invalid HDF5 files

A set of known invalid HDF5 files should be created or collected for this class of tests. The h5chk tool should reject them and list all the invalid parts or components it can detect.

4.2.1 Generating invalid HDF5 files

A set of "invalid file generators" should be written to create invalid HDF5 files. This may involve using the valid file generators to create valid files and then turn them into invalid files via some kind of binary file editors. Invalid data files can also be produced by using certain versions of HDF5 library that are known to be incompatible with the File Format.

5 Document Requirements

5.1 User Document

A man page style document should be provided as a user reference manual. It should describe the functions of the tool and all the command line options it supports. A similar abbreviated version of the same information should be coded in the tool to be display as a help page.

Another man page style document should be provided as a programming manual showing the definitions of the public API's of the h5chk library modules.

5.2 Tool Builder Document

A document should be provided to show the instructions of how to configure, build and execute the h5chk tool. The expected users of this document are system administrators.

5.3 Tool Tester Document

A document should be provided to describe the test data files and the instructions for running the test suite.

5.3.1 Test Suite Data

5.3.1.1 Valid File Generator

This contains a simple text file of instructions showing how to configure, build and execute the File Generators.

5.3.1.2 Real Life Production Files

A collection of valid HDF5 files collected from various production applications with a description for each file showing the following:

- A description of the production application (if privacy is desired, a generic description is acceptable)
- A brief description of contents of the data file such as approximate numbers of different HDF5 objects (datasets, groups, datatypes, …), features used (e.g., compressions) including any unusual features (e.g., user-defined filter).
- If known, the Version of HDF5 library and platform from which the file is created.

5.3.1.3 Invalid HDF5 data files

This consists of a set of invalid HDF5 files with a description for each file showing the following:

- A description of the process creating the invalid data file. If it is possible or practical, include the software tools or shell scripts that create the file.
- A brief description of contents of the data file such as approximate numbers of different HDF5 objects (datasets, groups, datatypes, …), features used (e.g., compressions) and a clear description the invalid components of the files.
- If appropriate, an identification of the version of HDF5 library and platform from which the file is created.

5.3.1.4 Overall Test Instructions

This consists of a text file of instruction describing steps to run the Test Suite and the expected behavior and output if tests run successfully.

6 Acceptance Requirements

6.1 Target Platforms

The software delivered should work properly in the following target platforms

- Linux (little endian)
- AIX (big endian)
- Microsoft Windows

6.2 Software to be delivered

6.2.1 The validation tool, h5chk

This consists of the source code files of the h5chk tool with auto-configuration setup such as configure, Makefile, etc, and the Tool Builder Document. It must be demonstrated that by following the instructions, one can build the tool in all Target Platforms.

6.2.2 Test Suite Data

This consists of the source code of the test file generators, the test data files and the Tool Tester Document. It must be demonstrated that by following the instructions, one can build the file generators, run them to generate test data files that are applicable.

6.2.2.1 Tool Test

It must be demonstrated that by following the instruction in the Tool Tester Document, one can run the Test Suite as expected in all Target Platforms.

Appendix

1 Possible extensions of h5chk

Two of the possible extensions of h5chk tool are described here.

1.1 H5repair

It corrects all errors found in the HDF5 file such that the file is valid again. This is not a simple extension and not all files can be repaired without a substantial loss of the file content. For example, a trivial but useless implementation of h5repair would be one that always repairs the corrupted file to contain just the root group.

1.2 H5recover

It salvages a corrupted or damaged HDF5 file by recovering as much file contents as it can. For example, it may discover isolated datasets which have no connected path with the root group. It will move them to a "lost+found" group, located in the root group of the file. Another example is that it discovers a block of raw data but cannot identify the dataset that owns it. It will move this block of raw data to the "lost+found" group and creates a new dataset to "own" this block of data. (It should be obvious that the author has borrowed much from fsck, the well known Unix file system repair tool.)
One difference between this tool and the h5repair above is that h5repair would delete isolate datasets while h5recover would move them to the lost+found group. It is also possible to combine the two into one tool of two modes.

2 References

[i] HDF5 File Format Specification as included in the HDF5 v1.6.5 release source tree, file name as doc/html/H5.Format.html