RFC: Replacing `H5Fis_hdf5()` with `H5Fis_accessible()`

Mohamad Chaarawi
Quincey Koziol

The HDF5 library provides a routine (`H5Fis_hdf5()`) to determine whether a given file is in the HDF5 format or not given the file name. This API routine is not robust however, and can be considered a bug in the API, because it only uses the default file driver when attempting to open the file. Furthermore, with the introduction of the HDF5 Virtual Object Layer (VOL) [1], all HDF5 files that have not been created through the native HDF5 VOL plugin will not be in the HDF5 file format, but they are still accessible from the HDF5 library. We propose to address the two concerns outlined above by adding a new routine, `H5Fis_accessible()`, to eventually replace `H5Fis_hdf5()` in the HDF5 library.

1 Introduction and Motivation

The HDF5 library provides users with a routine to check whethere a given file is in the HDF5 file format, and the HDF5 library is capable of opening it. This routine is defined as:

```
htri_t H5Fis_hdf5(const char *name);
```

When successful, this routines returns a positive value, for TRUE, or 0 (zero), for FALSE. On any error, including the case that the file does not exist, it returns a negative value.

This routine is not robust enough to check HDF5 files created by a driver different than the HDF5 default SEC2 virtual file driver (VFD). If an HDF5 file is created with a family file driver for example, the routine will incorrectly return false when invoked with the file name, even though the file is in the HDF5 file format.

A new feature that will make it into the HDF5 1.10 series is the ability to created HDF5 files that are accessible through the HDF5 API but with a different file format than the native file format. Another problem with the `H5Fis_hdf5()` function as it stands is that it would not handle such files/containers in non-native HDF5 format. The API currently does not provide an alternative for such a routine to work on files created by the non-native VOL plugin.

2 Approach

We propose to add a new routine to address the concerns detailed in the earlier section. The new routine will be defined as:

```
htri_t H5Fis_accessible(const char *name, hid_t fapl_id);
```

This routine checks whether a given file can be opened with the given file access property list (`fapl_id`). When successful, this routines returns a positive value, for TRUE, or 0 (zero), for FALSE. On any error, including the case that the file does not exist, it returns a negative value.

The additional `fapl_id` argument would be used to tell the library what VFD or VOL plugin to use to try and open the file to see if it is accessible. For example, the user could call `H5Pset_fapl_family()` on the `fapl_id` to check if a given file can be accessed with the family file driver.

We decided to change the name of the function rather than fix the old function by adding the `fapl_id` argument for backward compatibilty issues and the better representation of what the function does with the new name.

3 Implementation and Recommendation

A protoype implementation for `H5Fis_accessible()` is done and available for testing in the VOL branch located here: `http://svn.hdfgroup.uiuc.edu/hdf5/features/vol`. New tests have been added to verify that the routine works with the different VFDs that ship with the HDF5 library.

We recommend to add this new routine to the trunk and the 1.8 branch and deprecate `H5Fis_hdf5()` since the former is a more robust version of the latter.

Revision History

July 29, 2014 Version 1 circulated for comment within The HDF Group.

References

[1] Q. K. Mohamad Chaarawi, "RFC: Virtual Object Layer." `http://svn.hdfgroup.uiuc.edu/hdf5doc/trunk/RFCs/HDF5/VOL/2014-07-07-RFC_VOL.docx`, 2014.