RFC: Setting File Access Property List for accessing External Links

Vailin Choi
The HDF Group
9/4/2008

1. Purpose

When an external linked target (child) file is accessed, the HDF5 library uses the same file access property list as the main (parent) file to open the target file**.** The current implementation has two problems:

1. The library may fail to open the target file if the driver associated with the parent's file access property list is different from that of the target. For example, a *sec2* parent file versus a *family* target file.

2. The library does not provide the user with the flexibility of specifying a driver to use in opening the target file.

The first problem is filed as bug #1292 and will be addressed at a later time.

This document addresses the second problem which is filed as bug #1247. Two new public routines are added to the link access property so that the user can associate the file access property list identifier for the link access. The existing API for *File Access Properties* can be used in setting the appropriate driver for the target file.

There is no compatibility issue involved for adding these two public routines.

2. Modifications

1. The two new public routines for link access are:

hid_t
H5Pget_elink_fapl (hid_t lapl_id)
This routine returns the file access property list identifier that is set for the link access property list identifier, *lapl_id*. When no such identifier is set, this routine returns H5P_DEFAULT.

herr_t
H5Pset_elink_fapl (hid_t lapl_id, hid_t fapl_id)
This routine sets the file access property list identifier, *fapl_id*, to use when the target link associated with *lapl_id* is accessed. Please note that the setting is transient in nature.

2. Changes to internal routine *H5L_extern_traverse()*

This routine originally uses the parent's file access property list to open the target file. Changes are made to this routine to get the file access property list identifier that is set for the link access via *H5Pget_elink_fapl ().* If H5P_DEFAULT is returned*,* the same processing as before is done to retrieve the parent's file access property list identifier which is then used to open the target file. Otherwise, the file access property list identifier returned by *H5Pget_elink_fapl ()* is used to open the target file.

3. Documentation changes

   - Add description for *H5Pget_elink_fapl()* and *H5Pset_elink_fapl()* to the *Link Access Properties* group of the Property List Interface section in the reference manual.
   - Add description about the *opening* behavior of the target file to *H5Lcreate_external()* in the Link Interface section of the reference manual. Also add clarification about the parameter *lapl_id,* which holds properties used for traversal to the place where the link is to be created.

3. Use case

A user creates an external link from the main file to the target file. The main and target files might be of different file types. The user can set the file driver to use for the target file in the file access property list identifier. This identifier is then associated with the access property list via *H5Pset_elink_fapl().* When the external link is accessed, the library retrieves the file access property list associated with the link access via *H5Pget_elink_fapl()* and opens the target file with the specified driver.

Simplified code sample:

```
/* create the main file */
mfid = H5Fcreate(“main”, H5F_ACC_TRUNC, H5P_DEFAULT, H5P_DEFAULT);

/* create file access property for the target file to be a "family" file */
/* this can be other file types e.g. split, core, direct, multi, … etc. */
new_fapl_id = H5Pcreate(H5P_FILE_ACCESS);
H5Pset_fapl_family(new_fapl_id, FAMILY_SIZE, H5P_DEFAULT);

/* create the target file to be a "family" file */
tfid = H5Fcreate(“target”, H5F_ACC_TRUNC, H5P_DEFAULT, new_fapl_id);
gid = H5Gcreate2(tfid, "A", H5P_DEFAULT, H5P_DEFAULT, H5P_DEFAULT);

/* create external link from the main file to the target file */
H5Lcreate_external(“target”, "/A", mfid, "ext_link", H5P_DEFAULT, H5P_DEFAULT) ;

/* set file access property list for link access to use the family driver */
lapl_id = H5Pcreate(H5P_LINK_ACCESS);
H5Pset_elink_fapl(lapl_id, new_fapl_id) ;
```

```
/* open the target link */
oid = H5Oopen(mfid, "ext_link", lapl_id);
```

4. Testing

- Add test to *links.c* to verify that the external linked target file with physical layout different from the parent can be successfully opened. This test is repeated for *multi* file.

Create the main file (parent) file
Create the target (child) file to be a *family* file
Create external link from the main file to the target file and group via *H5Lcreate_external()*
Set the file access property list, *fapl*, to use the family driver via *H5Pset_fapl_family ()*
Set the file access property list for the link access to *fapl* via *H5Pset_link_fapl()*
Open the target linked group
Should succeed in opening the target link

- Add test to *links.c* to verify that processing done to the external linked target object is correctly handled when the parent and target files have the same physical layout but different access methods (i.e. drivers). This test is repeated for *stdio*, *direct* and *log* files with the appropriate processing.

Create the main file (parent) file
Create the target (child) file to be a *core* file, consisting of one dataset with late storage allocation
Get file size of the target file
Create external link from the main file to the target file and dataset via *H5Lcreate_external()*
Set the file access property list, *fapl*, to use the core driver via *H5Pset_fapl_core()* without backing store
Set the file access property list for the link access to *fapl* via *H5Pset_link_fapl()*
Open the target linked dataset
Write data to the dataset
Close the dataset
Close the file
Verify that file size of the target file is the same as before

5. Limitations

The new public routine *H5Pset_elink_fapl()* only sets the file access property list identifier to use for one target link access. When there are external links across multiple files which are of different file types, the following example illustrates how this new routine can be used to correctly open the target files:

```
/* create target file 1 to be a "core" file */
/* target1: /C/D/Dataset */
core_fapl = H5Pcreate(H5P_FILE_ACCESS);
H5Pset_fapl_core(core_fapl, CORE_INCREMENT, TRUE);
tfid1 = H5Fcreate("target1", H5F_ACC_TRUNC, H5P_DEFAULT, core_fapl);
```

```
gid1 = H5Gcreate2(tfid1, "C", H5P_DEFAULT, H5P_DEFAULT, H5P_DEFAULT);
gid2 = H5Gcreate2(gid1, "D", H5P_DEFAULT, H5P_DEFAULT, H5P_DEFAULT);
dcpl = H5Pcreate(H5P_DATASET_CREATE)) < 0);
H5Pset_alloc_time(dcpl, H5D_ALLOC_TIME_LATE) < 0);
dset = H5Dcreate2(gid2, "Dataset", H5T_NATIVE_INT, space, H5P_DEFAULT, dcpl,
        H5P_DEFAULT);
H5Dclose(dset);
H5Gclose(gid2);
H5Gclose(gid1);
H5Fclose(fid) ;
filesize = h5_get_file_size(filename1);

/* Create target file 2 to be  a “family” file */
/* target2: /A/B/ext1 → target1:/C/D/Dataset */
fam_fapl = H5Pcreate(H5P_FILE_ACCESS);
H5Pset_fapl_family(fam_fapl, FAMILY_SIZE, H5P_DEFAULT);
tfid2=H5Fcreate(filename2, H5F_ACC_TRUNC, H5P_DEFAULT, fam_fap);
gid1=H5Gcreate2(tfid2, "A", H5P_DEFAULT, H5P_DEFAULT, H5P_DEFAULT);
gid2=H5Gcreate2(gid1, "B", H5P_DEFAULT, H5P_DEFAULT, H5P_DEFAULT);
H5Lcreate_external(“target1”, "/C/D/Dataset", gid2, "ext1", H5P_DEFAULT, H5P_DEFAULT);
H5Gclose(gid2);
H5Gclose(gid1);
H5Fclose(fid);


/* create the main file */
/* main file:/ext2 → target2:/A/B */
mfid = H5Fcreate(“main”, H5F_ACC_TRUNC, H5P_DEFAULT, H5P_DEFAULT);
H5Lcreate_external(“target2”, “/A/B”, mfid, “ext2", H5P_DEFAULT, H5P_DEFAULT);

/* Set group access property list for link access to use the family driver */
gapl_id = H5Pcreate(H5P_GROUP_ACCESS);
H5Pset_link_fapl(gapl_id, fam_fapl);
gid = H5Gopen2(fid, "ext2", gapl_id);

/* Set dataset access property list for link access to use the core driver without backing store */
dapl_id = H5Pcreate(H5P_DATASET_ACCESS);
H5Pset_fapl_core(core_fapl, (size_t)CORE_INCREMENT, FALSE);
H5Pset_link_fapl(dapl_id, core_fapl);
did = H5Dopen2(gid, "ext1", dapl_id);
H5Dwrite(did, H5T_NATIVE_INT, H5S_ALL, H5S_ALL, H5P_DEFAULT, points);

H5Dclose(did)
H5Gclose(gid)
H5Fclose(fid)

/* verify that the file size for target1 is the same as before */
```

6. Future enhancement

The Link Interface allows powerful manipulation of links in HDF5. A user guide on this topic may provide users with effective usage of this API.