Programmatic Control of Dynamic Plugins

The loading of external dynamic filters can be controlled during runtime with an environment variable, HDF5_PLUGIN_PRELOAD, but this variable offers no option of control from within a program built on the library. The recommendation to change the state of the global plugin variable from negative logic, disable plugins, to positive logic, enable plugins is discussed in this document.

Introduced with

HDF5-1.8.15

in

May 2015

Copyright Notice and License Terms for HDF5 (Hierarchical Data Format 5) Software Library and Utilities

Contents

1. Introduction

The loading of external dynamic filters can be controlled during runtime with an environment variable, `HDF5_PLUGIN_PRELOAD`. However, it offered no option of control from within a program built on the library.

The environment variable can control the loading of dynamic filters at runtime, but it will disable it for all running programs that access that variable using the library. It is expected that the environment variable will be set before any programs execute and remain constant throughout the programs' execution life.

The need for finer grained control of the feature was exposed when HDF5 tools were built with the static-exec option and attempted to use dynamic filter loading. Because the tool and the filter used different runtime instances an exception was raised when a different C runtime tried to free the memory allocated by the other C runtime instance.

Another recommendation was to change the state of the global plugin variable from negative logic, disable plugins, to positive logic, enable plugins.

2. Use Cases

1. Disable all plugins - `H5PLset_loading_state (0)`
2. Enable all plugins - `H5PLset_loading_state (-1)`
3. Disable plugin X - requires user to negate the state with a 0 in bit position X and AND it with the result from a `H5PLget_loading_state` call.

```
H5PLget_loading_state(&curr_setting)
new_setting = curr_setting & ~H5PL_FILTER_PLUGIN
H5PLset_loading_state (new_setting)
```

4. Enable plugin X - requires user to set the state with a 1 in bit position X and OR it with the result from a `H5PLget_loading_state` call.

```
H5PLget_loading_state(&curr_setting)
new_setting = curr_setting | H5PL_FILTER_PLUGIN
H5PLset_loading_state (new_setting)
```

3. Implementation

In the H5PL.c file in the source folder, the local variable is declared as a signed int:

Source File: H5PL.c

```
static int          H5PL_plugin_g = -1;
```

This variable is used for both the global disabling all plugins as well as for the individual plugins. If all plugins should be disabled, the H5PL_plugin_g should be zero. If this is variable is non-negative then the value of the individual bits control the individual plugins. The plugin bit will correspond with the H5PL_type_t enum value for that plugin. For filters, bit 0 will enable filter plugins if set to 1.

In the H5PLextern.h file is the existing declaration of the H5PL_type_t enum:

Source File: H5PLextern.h

```
/********************/
/* Public Typedefs */
/********************/
/* Plugin type */
typedef enum H5PL_type_t {
     H5PL_TYPE_ERROR        = -1,  /*error                    */
     H5PL_TYPE_FILTER       = 0,   /*filter                   */
     H5PL_TYPE_NONE         = 1    /*this must be last!       */
} H5PL_type_t;
```

The functions use one argument to enable or disable the individual plugins. The function need to check the environment variable method of disabling the use of dynamic filter loading in order to not override the environment setting. The argument directly sets/unsets the global H5PL_plugin_g variable.

Source File: H5PL.c

```
/*-------------------------------------------------------------------------
 * Function:   H5PLset_loading_state
 *
 * Purpose:   Control the loading of dynamic plugins.
 *
 *        This function will not allow plugins if the pathname from
 *        the HDF5_PLUGIN_PRELOAD environment variable is set
 *        to the special "::" string.
 *
 *        plugin bit = 0, will prevent the use of that dynamic plugin.
 *        plugin bit = 1, will allow the use of that dynamic plugin.
 *
 *        H5PL_TYPE_FILTER changes just dynamic filters
 *        A negative value will enable all dynamic plugins
 *        A zero value will disable all dynamic plugins
 *
 * Return:    Non-negative or success
```

```
     *
     *-------------------------------------------------------------------------
     */
    herr_t
    H5PLset_loading_state(int plugin_flags)
    {
         char  *preload_path;
         herr_t ret_value = SUCCEED;   /* Return value */

         FUNC_ENTER_API(FAIL)

         /* check for global setting first */
         if(plugin_flags < 0)
             plugin_flags = -1;
         /* change the bit value of the requested plugin(s) */
         H5PL_plugin_g = plugin_flags;
         /* check if special ENV variable is set and disable all plugins */
         if(NULL != (preload_path = HDgetenv("HDF5_PLUGIN_PRELOAD"))) {
             /* Special symbol "::" means no plugin during data reading. */
             if(!HDstrcmp(preload_path, H5PL_NO_PLUGIN))
                 H5PL_plugin_g = 0;
             }
         }
     done:
         FUNC_LEAVE_API(ret_value)
    } /* end H5PLset_loading_state() */

    /*-------------------------------------------------------------------------
     * Function:   H5PLget_loading_state
     *
     * Purpose:   Query state of the loading of dynamic plugins.
     *
     *        This function will return the state of the global flag.
     *
     * Return:    Zero if all plugins are disabled, negative if all
     *            plugins are enabled, positive if one or more of the plugins
     *            are enabled.
     *
     *-------------------------------------------------------------------------
     */
     herr_t
     H5PLget_loading_state(int* plugin_flags)
     {
         herr_t ret_value = SUCCEED;    /* Return value */
         FUNC_ENTER_API(FAIL)

         *plugin_flags = H5PL_plugin_g;
     done:
         FUNC_LEAVE_API(ret_value)
 } /* end H5PLget_loading_state() */
```

To support the flags parameter, the following convenience defines help because the bit position defines in `H5PL_type_t` enum start at 0.

Define Value

```
   /* Common dynamic plugin flags */
    #define H5PL_FILTER_PLUGIN        0x0001
```

Also the `H5PL_no_plugin` function has been removed as it was never used. The only use of `H5PL_plugin_g` variable is in the `H5PL_load` function.

Private H5PL_load function in H5PL.c

```
const void *
 H5PL_load(H5PL_type_t type, int id)
 {
     htri_t      found;               /* Whether the plugin was found */
     const void  *plugin_info = NULL;
     const void  *ret_value = NULL;
     FUNC_ENTER_NOAPI(NULL)
     /* Check for "plugins are disabled" indication" */
     if(H5PLplugin_g == 0)
         HGOTO_ERROR(H5E_PLUGIN, H5E_CANTLOAD, NULL, "required dynamically
loaded plugin '%d' is not available globally", id)
     switch (type) {
     case H5PL_TYPE_FILTER:
         if((H5PL_plugin_g & H5PL_FILTER_PLUGIN) == 0)
              HGOTO_ERROR(H5E_PLUGIN, H5E_CANTLOAD, NULL, "required
dynamically loaded filter plugin '%d' is not available", id)
         break;
     default:
         HGOTO_ERROR(H5E_PLUGIN, H5E_CANTLOAD, NULL, "required dynamically
loaded plugin '%d' is not available", id)
     }
     /* Initialize the location paths for dynamic libraries, if they aren't
      * already set up.
      */
...
}
```

Public Header File

```
H5_DLL herr_t H5PLset_loading_state(int plugin_flags);
H5_DLL herr_t H5PLget_loading_state(int* plugin_flags/*out*/);
```

4. Revision History

May 2015	Version 1.