RFC: Bug Fix for SDstart for protected file
Date: January 27, 2005
Expires: January 31, 2005

Problem (see Bug 315):

> If SDstart() is called with DFACC_CREATE and the name of an existing file, but without write permission to the file and with write permission to the directory, SDstart will fail and the file will be deleted. (from Bug 315)

Analysis:

This bug is confirmed.

Proposed Fix:

The easiest fix is to check the file, and exit from `SDstart` before calling `ncreate`. Figure 1 shows the basic change in `SDstart` (in mfsd.c). Figure 2 shows proposed code to check the protection on the file.

```
       if(HDFmode & DFACC_CREATE)
        { /* create file */
+         if(!can_clobber(name))
+             HGOTO_ERROR(DFE_DENIED, FAIL);
          cdfid = nccreate(name, NC_CLOBBER);
        }
      else
```

Figure 1

```
+ int can_clobber(const char *name)
+ {
+ int res;
+ struct stat buf;
+
+     res = stat(name, &buf);
+
+     if (res < 0) {
+     return(1);
+     }
+
+     if ((buf.st_mode & (S_IWUSR | S_IWGRP | S_IWOTH)) == 0) {
+     return(0);
+     }
+     return 1;
+ }
```

Figure 2

A test for this case is shown in Figure 3. This test case illustrates the problem. In the current library, this test will fail. The second `SDstart` will return –1, but the second `fopen` will fail because the file is deleted.

```
#define FILE_NAME     "sdtest.hdf"  /* data file to test ID types */

extern int
test_sd()
{
    int32     fid;
    intn      status;
    mode_t mode;
    FILE *ff;
    intn      num_errs = 0;         /* number of errors so far */

    /* delete the file just to be sure */
    unlink(FILE_NAME);

    /* Create a file */
    fid = SDstart(FILE_NAME, DFACC_CREATE);
    CHECK(fid, FAIL, "SDstart");

    /* Close the file */
    status = SDend(fid);
    CHECK(status, FAIL, "SDend");

    mode =  S_IRUSR;
    status = chmod(FILE_NAME, mode);

    /* Create a protected file */
    fid = SDstart(FILE_NAME, DFACC_CREATE);
    VERIFY(fid, FAIL, "second SDstart");

    ff = fopen(FILE_NAME, "r");
    CHECK(ff, NULL, "fopen");

    if (ff != NULL) {
          fclose(ff);
    }

    /* Return the number of errors that's been kept track of so far */
    return num_errs;
}
```

Figure 3

Summary and Requested Action

This fix appears to be a simple and effective response to Bug 315. It does not address similar problems that might exist in other parts of the library. *Please review this proposal and raise issues as soon as possible.*

All the calls used are POSIX, so they should work on windows. I have not tested windows, though.