#SASGF

VIRTUAL

SAS® GLOBAL FORUM 2021

AMERICAS | MAY 18 – 20 ASIA PACIFIC | MAY 19 – 20 EMEA | MAY 25 – 26

1123-2021 How to Deal with Locked Datasets on UNIX

Kurt Bremser, Allianz Technology Austria

Learned to code at Technische Universität Wien

Have been working with SAS since 1998

Led the transition from Mainframe SAS to UNIX

Current position: Data Warehouse Administrator and Developer

Super User at the SAS Communities

Author of the Maxims of the Maximally Efficient SAS Programmer

The Problem

```
27          data target.ds;
28          set source.ds;
29          run;

ERROR: Keine Sperre verfügbar für TARGET.DS.DATA.
ERROR: Sperre vorhanden durch Prozess 12345678.
NOTE: The SAS System stopped processing this step because of errors.
WARNING: The data set TARGET.DS was only partially opened and will not
be saved.
```

The Problem

The Causes

- A dataset already exists in a “public” (shared) library
- It needs to be overwritten
- One or more users/processes have this dataset opened, for viewing, or as a source in a PROC or DATA step
- Because of the dread god Finagle, and his mad prophet Murphy, this happens in an important batch job that has a long chain of other processes following, so you get called up at 1 a.m.

Possible Remedies

- Set up a dedicated time window for updates; this implies that you need to stop your services, disconnect all current clients, possibly losing data or interrupting long-running processes (think a scheduled project in Enterprise Guide).
  This might not be a viable option for people who want to have 24/7 operation to accommodate the connected world.
- Search for the offending process and kill it, or contact the owner and ask him/her to close the dataset.
  Still, you need to rerun the job, and you might notice that still another process has the dataset open. Rinse and repeat.

Possible Remedies

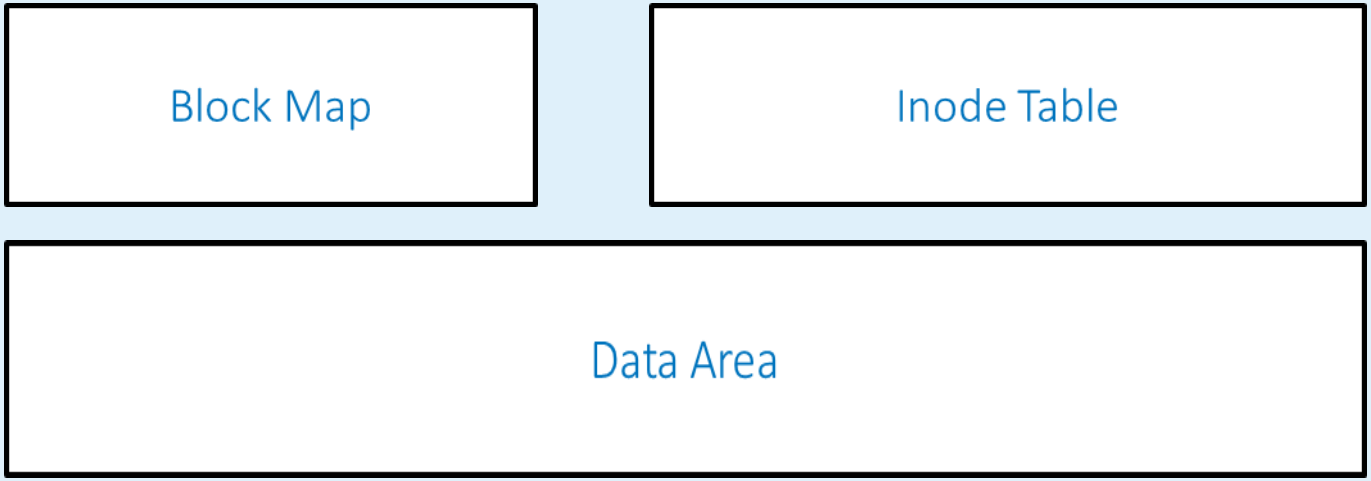
- Or delete the dataset *file* from the library (directory) before writing it.
  This is possible because of the way UNIX (and this, of course, includes Linux) structures its filesystem.

The following slides will show how this is done.

Background

The UNIX Filesystem

A UNIX filesystem consists of three major areas:



This is, obviously, a rough outline, as filesystems have developed over the years.

Background

The UNIX Filesystem

The block map keeps track of allocated and free blocks in the data area.

The inode table is an array of structures (inodes as in Indexed Nodes) that contain all the file's metadata, but **not** the file names.

Directories are files in the data area, containing only file names and associated inode numbers.

Inodes keep a count of links (directory entries) pointing to them, as it is possible to have several directory entries pointing to the same inode, or even none.

Background

The Windows Side

Although NTFS is a far cry from the old FAT file system on which Windows was initially based, it still does not allow to manipulate (remove or rename) a directory entry while the file is in use.

As a (tested) example, even with the **`/f`** flag, the **`del`** command will refuse to remove a file kept open by an application.

How It Works

Techniques Used

- Either external commands (if XCMD is enabled) or the FDELETE function can be used
- The external command used is **`rm -f`**; the **`-f`** prevents a request for confirmation in certain cases (no write permission on the file itself)
- The remove will work as long as the user issuing it has write permission on the directory (library)
- Either a data step or macro code is used for the call of the external command or the function

How It Works

The Macro Code

```
%macro ds_delete(ds=,ext=Y,safe=N);

/* ds_delete, Version 0.9
   This macro is designed to delete a dataset or view, in order to prevent a
   "Dataset is locked" message. It will automatically determine the physical path
   to the library and the proper filename extension. Depending on the presence
   of additional parameters, pure macro or data step code with external commands or
   the FDELETE function will be used.
*/

%if %length(&ds.) lt 1
%then %do;
  /* issue a usage note */
  %put Usage:;
  %put %nrstr(%ds_delete)(ds=dataset|view[,ext=y|n][,safe=y|n]);
  %put ds=dataset|view .. name of the dataset or view, single level considered in WORK;
  %put ext= (optional) .. Y will use external commands;
  %put safe=(optional) .. Y will cause macro code to be used (otherwise data step);
  %return;
%end;
```

How It Works

The Macro Code

```
/* make our macro behave gracefully */
%local
  dsid
  type
  suffix
  pathname
  fname
  fref
  command
  rc
;

/* crude check for validity of dataset name */
%if %sysfunc(countw(&ds.,.)) gt 2
%then %do;
  %put Too many levels!;
  %return;
%end;
```

How It Works

The Macro Code

```
/* expand single-level dataset name */
%if %sysfunc(countw(&ds.,.)) lt 2 %then %let ds = WORK.&ds.;

/* check if the dataset/view actually exists */
%let dsid = %sysfunc(open(&ds.));
%if &dsid. = 0
%then %do;
  %put Dataset or view does not exist!;
  %return;
%end;

/* determine physical filename suffix */
%let type = %sysfunc(attrc(&dsid.,MTYPE));
%if &type. = VIEW
%then %let suffix = sas7bvew;
%else %let suffix = sas7bdat;

/* get physical name of the directory */
%let pathname = %sysfunc(pathname(%sysfunc(attrc(&dsid.,LIB))));

/* build the complete filename */
%let fname = %lowcase(%sysfunc(attrc(&dsid.,MEM))).&suffix.;
```

How It Works

The Macro Code

```
/* external command or FDELETE */
%if %upcase(&ext.) = Y
%then %do;

  /* build the command */
  %let command = rm -f &pathname./&fname.;

  /* run the command, either with %SYSEXEC or FILENAME PIPE */
  %if %upcase(&safe.) = Y
  %then %do;
    /* this makes the code "safe" for use anywhere in code,
       by using only macro statements */
    %sysexec &command.;
  %end;
  %else %do;
    /* retrieve all responses (including stderr) and write them to the log */
    data _null_;
    infile "&command. 2>&1" pipe;
    input;
    put _infile_;
    run;
  %end;

%end;
```

How It Works

The Macro Code

```
%else %do;
  %if %upcase(&safe.) = Y
  %then %do;
    /* macro statements */
    %let rc = %sysfunc(filename(fref,&pathname./&fname.));
    %if &rc = 0
    %then %do;
      %let rc = %sysfunc(fdelete(&fref));
      %put &=rc.;
      %let rc = %sysfunc(filename(fref));
    %end;
  %end;
  %else %do;
    /* data step */
    data _null_;
    length fref $8;
    rc = filename(fref,"&pathname/&fname.");
    rc = fdelete(fref);
    put rc=;
    rc = filename(fref);
    run;
  %end;
%end;
%let dsid = %sysfunc(close(&dsid.));
%mend;
```

How It Works

The Technical Details

Removal of a directory entry does not (immediately) remove the file; only when the last link for an inode is removed, and no open file handle exists, will the file be de-allocated and the inode removed.

As long as a user has the file open, the contents (pre-update) will stay available; to see the updated dataset, the user has to close and re-open it.

This may also cause excessive use of storage while both versions exist physically.

How It Works

A Welcome Side Effect

There is an additional benefit:

If you do not have write permission on the file itself (for instance because the existing dataset was created by another user, and the default umask denies write permission to anybody else), you can still remove it and write your own dataset.

This can come in handy when working with shared libraries where several users are permitted to create datasets.

How It Works

Limitations

Appending in place is not possible. You need to concatenate the data in WORK, use the macro, and copy the resulting dataset to the final location.

Index files are not removed; this is not much of an issue, as SAS removes those files on its own when the new dataset is written.

It is also not intended for use on generational datasets.

Care must be taken with data on network shares, as the file server might not support all UNIX features.

Conclusion

Recommended Reading

For those interested in more in-depth information, I recommend the Wikipedia articles on the UNIX File System (https://en.wikipedia.org/wiki/Unix_File_System)

and on the inode (https://en.wikipedia.org/wiki/Inode)

Conclusion

Usage and Improvement

If you encounter problems, or find a useful improvement, feel free to comment on the SAS Communities Library Article **Avoiding the dreaded "Dataset is locked" Error** (https://communities.sas.com/t5/SAS-Communities-Library/Avoiding-the-dreaded-quot-Dataset-is-locked-quot-ERROR/ta-p/653210).

Thank you!

Kurt Bremser
https://communities.sas.com/t5/user/viewprofilepage/user-id/11562