Menage a Trois: UNIX, SCL, and SAS/SHARE!

Marge Scerbo
University of Maryland at Baltimore

Background:

In the age of client-server applications, gaining access to SAS/AF and SAS/SHARE on UNIX platforms is a major step into the future. SAS Institute has provided software for UNIX platforms since 1988. SAS Applications Facility (SAS/AF) Screen Control Language (SCL) was introduced in 1989 in version 6.03 of the SAS software for the DOS environment. SAS/SHARE has been available in version 5.16 on MVS since 4th Quarter 86 and on CMS since 1st Quarter 87, but until recently was not available for UNIX. Finally, with the 4th Quarter 93 Version 6.09 release for Family I UNIX platforms, all three are in the same place at the same time, with SAS/SHARE being available in beta release. At the present time, access to UNIX, SCL and SHARE is limited. As more sites move to these options, the possibilities of uses will grow.

Implications:

Platforms running the UNIX operating system are spreading rapidly across the nation. Introduced in the late 1960's at Bell Laboratories, the UNIX operating system is easily accessible, portable, and cheap. These systems provide a friendly environment which encourages users to share their files. With the drop in hardware costs and upgrades to the operating system simplifying installation and use, porting systems to UNIX platforms has become fairly easy and inexpensive. This is particularly true of applications developed in SAS Software; multivendor architecture allows for a system which has been developed on one platform to be converted and reinstalled on a UNIX machine. To a flexible systems developer, the move to an appropriately configured UNIX machine should provide an increase in the speed of development time and user response, while the process of learning the UNIX operating system should not prove too great a challenge.

The speed of programming, compilation, and access times under a UNIX operating system, as well as the X-windows interface to UNIX machines, should increase programmer efficiency and reduce development time. Compiling an SCL program on a mainframe can be painful and the response time for users even worse. Screen Control Language is a much more viable option on a UNIX machine than on an IBM running CMS!

SAS under UNIX allows various system options for accessing catalogs and datasets; access can be limited to single user read and/or write (FILELOCKS = FAIL) or allow for multiple read-write access (FILELOCKS = NONE). When FILELOCKS is set to FAIL, no concurrent access to either datasets or catalogs is allowed. In order to have a multiuser system, each user's individual directory must contain each catalog to be accessed; each time a catalog is updated it must be copied to all users' directories. When working on a system with FILELOCKS set to NONE, only one copy of a multi-user catalog is needed but there is the possibility of corruption. This corruption can occur not only when concurrently accessing the same observation in a dataset, but by simply allowing multiple concurrent updates to a dataset or catalog. Datasets are fairly easy to rebuild through system prompts, although some data may be lost. Catalogs are not as easily rebuilt, depending on the extent of the corruption.

These problems are dispensed with SAS/SHARE which allows multiple concurrent access, dependent on the type of file being accessed and on the system configuration. SAS/SHARE software allows for the development of applications which require concurrent access to SAS files (two or more users can simultaneously access a member of a SAS data library). Each user attached to a SAS/SHARE server can individually submit a request for data from the server.

SAS/SHARE allows for three different access mechanisms to shared data: single host, networked, and cross architecture. Single host access can be described as the users/clients and the server operating on the same physical machine. Access can also be networked whereby the server is running on one UNIX processor and the users executing on another...
UNIX platform. The third access mechanism is cross architecture. For example, users attached to a UNIX machine can access shared data actually stored on an MVS mainframe system. The network of shared data expands as needed!

**Catalogs:**

Realizing the virtually endless opportunities these three parameters offer, the next step is to utilize them efficiently and effectively. After the SAS/SHARE software has been installed properly on the system and the server is processing in the background, datasets and catalogs to be shared should be moved to or created in the server directory, whether in the root or in subdirectories. Permissions should be set appropriately; SAS/SHARE does not override UNIX level security.

To test if a catalog can be concurrently accessed, perform multiple AF commands to a module within the catalog. The AF command allows for display of AF applications while preventing write access to the AF catalog or its members. Multiple users should be able to access the catalog by performing an AF command from either the Display Manager Command line or from a batch program:

```
af c = cathtest.umcath.mainmenu.program
```

If SAS/SHARE has been configured correctly, multiple readonly access should be successful.

The next step is to test multiple PROC BUILD sessions from the program editor of Display Manager or from a batch program. PROC BUILD is a procedure in SAS/AF software which allows read-write access to a catalog or members of a catalog. Use the command:

```
proc build c = cathtest.umcath;
run;
```

The second concurrent attempt to execute this statement should fail and the user should be placed in browse mode. A lock will be placed on the catalog; multiple write access to catalog should fail. This will prevent damage to the catalog; concurrent access could cause corruption. Only one PROC BUILD session to a specific catalog should be permitted.

**LIBNAMES:**

The SAS code must contain appropriate LIBNAMES to the server catalog. It is good practice to store these entries in the autoexec file, the main menu program or initial program entry of the catalog so that the libnames reside in one place and can easily be accessed and modified if necessary. A libname statement which allows concurrent write access to the dataset might be:

```
rc = libname('ssd','remote',
     '/paper/crcserv/data',
     'server=crcserv sapw = secret');
```

where

- **ssd** associates the reference 'ssd' as a valid logical name for the SAS data library
- **remote** defines the engine that enables a user's SAS session to communicate with the server and thus access shared data
- **paper/crcserv/data** names a valid server subdirectory which stores the shared datasets
- **server** identifies the SAS server
- **sapw** states the password for the server.

Although this option is specified in the LIBNAME statement, it does not define the SAS library to be accessed; it merely establishes communication to a SAS server which has a UAPW= option in effect.

If there are datasets which will be opened in read-only mode, the LIBNAME would be:

```
rc = libname('lis','remote',
     '/paper/crcserv/list',
     'server=crcserv,sapw = secret',access = readonly);
```

where

- **lis** associates the reference 'lis' as a valid logical name for the SAS data library
- **remote** defines the engine that enables a user's SAS session to communicate with the server and thus access shared data
- **paper/crcserv/list** names a valid server subdirectory which stores the shared datasets
- **server** identifies the SAS server
- **sapw** states the password for the server.

**access = readonly** allows only read access to specified catalogs and datasets.
Sample System:

For an example, a system (UMCATH, copyright 1994 Scerbo/Ziskind) was developed for a cardiac catheterization clinic which tracked the records of patient visits, procedures, results and outcomes. Each record in the 50-some datasets is identified by three separate variables: history number, visit date and visit time (a patient may be seen more then once each day). A dataset, PATINDEX, acts as the index for the many datasets in the system. In addition to the three above-mentioned variables, patient last name is stored in this dataset, which is used for searching the database for specific records, producing extended tables for choices, and prevention of duplicate records. Assume that each time an observation is to be added, updated, deleted or reported upon, an AF screen is displayed requesting one or all of the identifiers. No observation can be selected until all three identifying variables are complete. When a user selects a specific record, that record is 'locked' in the dataset to protect from multiple access. The screen might look like:

```
&ERRMSG1__________________________
Enter as much information as available. Complete any or all of the fields: History Number, Date Entering Lab, Time Entering Lab, Patient Last Name and/or the first letter of Patient Last Name. If ALL fields are not completed, a list of appropriate choices will be displayed.

History Number: &HISTNO
Date Entering Lab: &VISTDAT
Time Entering Lab: &TIME
Patient Last Name: &PATLNAME
```

The command line of the screen has been turned off, and an error message line, ERRMSG1, has been added to return messages to the user.

The underlying SCL code to search for the appropriate record and lock it might be:

```scl
&ERRMSG1 = _blank_;
if &rc = %sysrc(_swnopd) then
  &errmsg1 = 'Record already in use!';
else if &rc = %sysrc(_swnowhr) then
  &errmsg1 = 'Record not found!';
if &rc ^= 0 then do;
  alarm;
  cursor histno;
  histno = _blank_;
  vistdate = _blank_;
  visttime = _blank_;
  patlname = _blank_;
  call close(&dsid);
  return;
else do;
  "process proceeding menus and programs;"
  call close(&dsid);
end;
where
&dsid = open('ssd.patindex','u');
*The dataset, PATINDEX, is opened in update mode ('u'). The dataset identifier used for further processing is stored in the variable, DSID. The libname, SSD, has already been defined as a dataset stored in a shared library.

call set(&dsid);
*The set statement allows the Dataset Data Vector (DDV) and the Screen Control Data Vector (SDV) to interact.

clause1 = 'histno = "lIhistnoll"';
clause2 = 'AND vistdate = "lI vistdate;
clause3 = 'AND visttime = "lI visttime;
rc = where(&dsid,clause1,clause2,clause3);
*Three separate WHERE clauses are defined and then the dataset is subset by a series of pre-built WHERE clauses.

&rc = fetch(&dsid);
*The observation is fetched from the subsetted dataset and moved into the DDV. The return code of the fetch operation is stored in a variable, FRC. A successfully
fetched observation will set this variable to 0 and lock the subsetted observation in the dataset while end-of-file markers (signifying no match has been found) will set the variable to -1. All other non-zero responses indicate possible problems in the operation.

```plaintext
errmsg1 = _blank_;  
* The errmsg1 variable is reset to null.

if frc = %sysrc(_swnoupd) then  
  errmsg1 = 'Record already in use!';
else if frc = %sysrc(_swnowhr) then  
  errmsg1 = 'Record not found';
if frc ^= 0 then do;  
  *If the return code stored in FRC is not equal to 0, either the observation could not be found or the observation is presently in use. The operation is sent back to the user with appropriate messaging and empty fields. Compare the variable FRC with the system macro variable %SYSRC and the mnemonics associated with the FETCH operation.
  *process proceeding menus and programs;
  call close(dsid);
end;
*Process any the datasets to be added/updated/deleted which the user has chosen. When complete, close the dataset, PATINDEX, which will remove the lock on the specific observation.

These few lines of code do a great deal of work. The dataset is first opened and subset to a specific observation. This observation is tested for existence and ownership. If the specific record is subset from the dataset and the FETCH operation is successful, this observation will be locked. The dataset will remain open on this record until functions on all identically identified datasets are completed. Since this one dataset is accessed for add, update, delete, and printing of individual patient records, the selection and lock of a particular 'record' means that all datasets with the same identifiers are locked and only up-to-date information is provided on a particular patient record.

System Return Codes:

According to SAS Institute, these lines can be reused across platforms by using the system return codes for the FETCH operation. The SCL functions performed return system return codes in the form of the %SYSRC macro. Various mnemonics specify the result of the operation performed. The return codes for these macro variables consistent across platforms. For instance, each time the SCL FETCH function is performed, two different %SYSRC macros may be returned: _SWNOUNDP which if returned states that the record cannot be updated at this time because of a pending lock and _SWNOWHR which if returned states that the record cannot satisfy the WHERE clause and thus cannot be fetched. Again, use of the %SYSRC macro variable and its parameters verifies whether the FETCH function was successful; if the FETCH is successful, the observation is locked until all editing is completed (or the user chooses to cancel the process) and the dataset is closed. Therefore, multiple access to the dataset is allowed but not concurrent access to a particularly identified record.

Summary:

This somewhat condensed code allows safe concurrent access to properly defined shared datasets while providing locking of specific records while adding, updating, deleting or printing. By using SAS/AF and SAS/SHARE together, the SAS Software system becomes more DBMS (Database Management System) like. Combining these options on a UNIX machine adds to ease and speed of use by both programmers and users of the system.

Other Operating Systems:

SAS/SHARE is available on VMS machines in beta versions. For DOS users, there is no mechanism for sharing data; most DOS users perform individual analysis on read-only data, but cannot perform multiple write access. SAS/SHARE is available under both OS/2 and Windows NT in experimental versions. Version 6.08 SAS for Windows allows users to have concurrent read-write access only to datasets through SAS/CONNECT to machines running SAS/SHARE. Windows is not a multitasking operating system so concurrent access to catalogs is not possible. According to SAS Institute, SAS/SHARE software that is presently experimental or beta will, hopefully, be production in 1994.
Special thanks to Greg Shanker, SAS Institute, for his assistance

References:


