

You Want Me to Move How Many Thousand Files from MVS to UNIX?

Kenneth A. Hardy

Arturo Barrios

Sally Muller

University of North Carolina, Chapel Hill, NC

Abstract

The University of North Carolina at Chapel Hill's Institute for Research in Social Science maintains one of the largest social science data warehouses in the country. Its collection of over 25,000 files, totaling over 225 gigabytes, has been stored on disk and tape on the campus's academic IBM mainframe for many years. In January of 1995, UNC's Office of Information Technology announced that all IBM/MVS mainframe services would be withdrawn from public use at the end of December, 1995 and that central computing services would shift to a distributed UNIX environment. This meant that the entire archive had to be migrated from the IBM mainframe to an E-mass Datatower running FileServ software on a Convex 3480 before the 1995 fall semester.

The task was complex because file sizes ranged from a few hundred bytes to over 2 gigabytes. Many files were in formats such as SAS V5.18 format libraries, early version SPSS system files and OSIRIS data dictionaries. In addition, several thousand files included IBM data types not recognized in the UNIX environment such as IBM column binary or packed decimal. To complicate matters more, a substantial portion of the files were stored on 3480 cartridge tapes and there was not sufficient disk storage to stage them to disk prior to transfer. Clearly, just using FTP wasn't going to work.

This paper shows how we used base SAS® and SAS/CONNECT® for automating the file conversion and transfer process for disk and tape files, and for checking that all the files had been moved. We also show how files containing data types unique to IBM mainframes can be transferred and read by SAS in a UNIX environment.

The Problem

The University of North Carolina's central computing organization, the Office of Information Technology, announced in January of 1995 that by December of 1995 the IBM mainframe, which serviced hundreds of campus users, would be decommissioned and that a UNIX based system running on a Convex C3480 would become the primary statistical and general purpose, central computing platform for the campus. The Institute for Research in Social Science had a large social science data archive consisting of over 25,000 files totaling approximately 225 gigabytes that would have to be migrated from the IBM platform to the UNIX platform so that students and faculty could continue to use the data in the archive for research and assignments.

We could not simply FTP all our files from the IBM mainframe to the UNIX platform for a number of reasons:

The files had a variety of formats and a substantial proportion of these files would need some form of conversion prior to transfer. The file formats included EBCDIC text and numbers, Version 5 and Version 6 SAS libraries, SAS Version 5 and Version 6 format files, several versions of SPSS files, OSIRIS data and dictionary files, column binary data and variable length files containing System 370—specific data types, such as packed decimal.

While about twenty-two percent of the files were stored on online disk and sixty percent were migrated into the Hierarchical Storage Management (HSM) system, eighteen percent were stored on 3480 tape cartridges in a STK tape silo managed by CA-1. The latter were generally quite large (some over 2 gigabytes) and because they were on tape, they were not directly accessible by our version of FTP. To make matters worse, not enough spare online disk

space was available to routinely copy these very large, tape files online for FTP transfer.

The sheer number of files to be moved also meant that if we were going to make our deadline, we needed automated tools for moving large numbers of files in a single transfer operation and for verifying that files were transferred from one platform to the other. Luckily, we found that we could use base SAS to automate most of the conversion process and SAS/CONNECT and FTP to perform the transfers using batch jobs on the IBM mainframe.

The Solution

Here are the steps we followed. Each step will be discussed in detail.

- 1) Identify and list a group of files for transfer
- 2) Recall any HSM migrated files
- 3) Determine file format of each file
- 4) Determine storage device of each file
- 5) Convert files needing conversion
- 6) Determine appropriate UNIX subdirectory for each file and create the appropriate subdirectory on the UNIX platform
- 7) Transfer the files from the IBM to the UNIX system using SAS/CONNECT or FTP.

Step 1 -- Identify and List a Group of Files for Transfer

The data set names (DSNs) of all files needing transfer were cataloged in the system catalog. Since they followed a fairly straightforward naming convention, we could easily identify groups of files that were likely to have a known set of file formats. For example, we had over 7000 files with DSNs that were similar to IRSS.ROPER.Snnnn. These files contained data acquired from the Interuniversity Consortium for Political and Social Research.

Once we had identified a group of files for transfer, we needed a master list of DSNs along with volume information and, for online files, DCB information in a file that we could then process with SAS.

Initially, we investigated creating a listing of DSNs with ISPF under TSO but we found the TSO DSAT command was much superior for this purpose. Figure 1 shows the TSO batch job containing the

DSAT command that we used to create the listings.

Figure 1
Example Code for Batch TSO Job for Listing DSNs with TSO DSAT Command

```
//DSATLIST JOB UNC.C.P4505,HARDY.K,TIME=5,PRTY=9
// EXEC BATCHTSO
//SYSTSIN DD *
ALLOC FI(DSAT) DSN(BIGMOVE.DSALIST)
DSAT 'IRSS.ROPER' HARDCOPY(DSAT)
FREE FI(DSAT)
```

Figure 2 contains a small sample of the records found in the file created by using the DSAT command. Notice that the TSO DSAT command produces statistics on the file's size in tracks, create-date, last-accessed-date, block size, logical record length, and data set organization.

Figure 2
Sample of Records Found in File Created by TSO DSAT Command

SERIAL	ALLO	USED	EX	DSORG	DCB	ATTRIBUTES	CR. DATE	DSNAME
INGRAT								IRSS.ROPER.ASAR90.CDSK
107648								IRSS.ROPER.ASAR90.DATA
107648								IRSS.ROPER.GIS72RLDATA
INGRAT								IRSS.ROPER.GIS72RLSPCRD
UDAT88	4	4	1	A-FI	FILE	4000	00	02/23/87
	IRSS.ROPER.S0148G.CDSK							
UDATE2	8	8	1	A-FI	FILE	1000	170	02/23/87
	IRSS.ROPER.S0148G.DATA							
UDAT00	1	1	1	A-FI	FILE	1000	00	02/23/87
	IRSS.ROPER.S0148G.DNCT							

Step 2 -- Recall HSM Migrated Files

To avoid interrupting programs that were converting or transferring data files, we "forced" retrieval of all migrated files scheduled for transfer before we began processing them. Otherwise each time HSM retrieved a migrated file for processing it would interrupt a file transfer program. We either used a series of ISPF panels for this purpose or a custom written TSO CLIST, which could retrieve all DSNs under a specified top—level DSN stem. Due to disk space constraints, we could not always recall all the files under a two-level stem but very often could recall and process several hundred files at a time.

Step 3 -- Determine File Formats

Identifying the file formats of a large group of files could be done in two ways — with "good" file naming conventions and without good file naming conventions.

Fortunately, we had adhered to a reasonably systematic file naming convention during the 30 years that we added files to the archive. It was therefore possible to write SAS code to process the listings of files, generated in step 1, and use certain character strings in the DSNs as indicators of file type. For example, DSNs ending in "SAS" were always SAS data libraries; those ending in "DATA" were almost always simple EBCDIC files; and those ending in "SPSS" were SPSS system files. Our naming conventions were not perfect but they worked for about 99 percent of our files.

In the absence of good naming conventions, determining file formats is possible but much more difficult. The first requirement is that all the files are cataloged and stored online disk so that the TSO DSAT command can obtain DCB information. With this DCB information and prior knowledge of common file formats, it is possible to make a "best guess" about the type of a file. We include the DCB information for a variety of file formats found in our environment in Appendix A. This approach will work fairly well but it is not foolproof and can leave a large number of unidentified file formats that must be handled individually.

Step 4 -- Determine Storage Devices

The file listings produced in Step 1 also provided information about each file's storage device. We used this information to determine how to process the file. The VOLSER of the files storage device is found in the leftmost column. At our site any file with a volume serial number beginning with "UDAT" was located on disk. Files with a VOLSER of "MIGRAT" needed to be recalled. Files with a six digit VOLSER were located on 3480 tape.

Step 5 -- Convert Files Needing Conversion

While most of our files did not require any special treatment or conversion prior to transfer, some file types did require preprocessing.

SAS Version 5.18 format libraries had to be read with PROC VSTOV6 and converted to SAS Version 6 catalog entries prior to transferring them with PROC UPLOAD. Example code for this process

can be found in Appendix C.

SPSS Version 4 and SPSSX system files needed to be read with SPSS Version 4 and converted to SPSS export format files. A sample program in Appendix D illustrates converting an SPSS system file to an SPSS portable file. They then could be transferred with FTP as character (ASCII) files.

SPSS system files created by pre-SPSSX versions of SPSS could not be read by SPSS Version 4. They could, however, be processed in two steps: 1) Read the SPSS file with SAS with the SPSS I/O engine and create a SAS files (see appendix E). 2) Read the SAS file with SPSS Version 4 using the GET SAS command and create an SPSS export format file.

Text partitioned data sets (PDS) had to be processed so that each member was made into a separate MVS file. (SAS/CONNECT can move a PDS as a single file, but will truncate records longer than 80 bytes.) Therefore, we resorted to creating a file for each member and then transferring these files.

Step 6 -- Determine and Create Appropriate UNIX Subdirectory

We decided that we needed to organize subdirectories on the UNIX platform as some straightforward translation of the IBM DSNs. This would enable users accustomed to DSNs to locate files easily. It also provided us with a way to reconstruct DSNs from a UNIX subdirectory so we could check lists of UNIX files against lists of MVS files to ensure that we transferred everything.

For most files we could use the second and third level of the DSN to create subdirectories under our main data storage subdirectory on the UNIX platform. We could use the fourth through last levels of the DSN connected with periods to create a filename. We also decided that all subdirectory and file names would be lowercase. So, for example, the DSN:

IRSS.ICPSR.S1234.YR1994.DATA

would translate to:

/pub/irss/icpsr/s1234/yr1994.data

The SAS program in Appendix B shows SAS code for accomplishing this.

Step 7 – Transfer Files

We knew that using FTP or SAS/CONNECT in interactive mode to transfer thousands of files would be far too time consuming so we did a little research and found that both of these file transfer methods could be executed as batch jobs under MVS.

Moreover, a little experimentation revealed that transfers initiated on the IBM (uploading) to the Convex were more efficient than those initiated on the Convex (downloading). Batch processing also allowed us to convert and transfer several hundred files overnight when the two machines and the communication link between them were less busy with users' tasks.

When we began, we used only SAS/CONNECT to transfer files. We soon found out that it worked well for SAS libraries and format catalogs and, at our site, was the only way to directly transfer files stored on tape. But for other files on disk it was much slower than FTP. In particular, for transferring simple EBCDIC files, files of column binary data with fixed length records, or any other files that required simple binary transfer, we found FTP to be much faster. Since we had thousands of EBCDIC and column binary files, we modified our code to use PROC UPLOAD only when necessary and FTP as the default.

Basics for Batch SAS/CONNECT and FTP

We've listed and discussed each of our steps as though they were discrete and independent of each other. In practice, steps one and two were done separately and steps three through seven were usually included in a single batch job that processed the list of files created in step one. We'll turn to an example of an entire job in the next section but first we want to show you parts of jobs that use SAS/CONNECT and FTP in batch mode.

Figure 3 shows you some of the code we used to create subdirectories and upload several basic file types using SAS/CONNECT's PROC UPLOAD. Note that to use SAS/CONNECT in this manner, you must have it installed on both the sending and receiving machines. In this example, we show how to upload basic EBCDIC files, simple binary files, SAS files, and SAS Version 6 format catalogs, so you can get a grasp of how PROC UPLOAD handles various

file formats. We found that these four types of uploads would transfer every type of file that we needed to transfer.

Figure 3
Code for Batch SAS/CONNECT File Transfer of Various File Types From IBM to UNIX

```

=====
*** Necessary connection information for SAS/CONNECT ***
=====
%LET REMOTEIP=152.2.25.2; /* Default remote net address */

OPTIONS COMAMID = TCP /* Communications protocol */
REMOTE = REMOTEIP; /* Default system for SIGNON */

FILENAME RLINK /* Default SAS/CONNECT script */
YOUR.SAS.CONNECT.SCRIPT;

=====
*** Signon to remote platform ***
=====
SIGNON;

=====
*** Create a subdirectory & transfer EBCDIC file ***
*** Also works for SPSS portable format files. ***
=====
IF type='EBCDIC' THEN DO;
RSUBMIT;
X 'mkdir -p /sub1/sub2/sub3';
PROC UPLOAD INFILE='SIMPLE.EBCDIC.DSN'
OUTFILE='/sub1/sub2/sub3/file.name'
STATUS=NO;
RUN;
ENDRSUBMIT;
END;

=====
*** Create a subdirectory & transfer binary copy of file ***
=====
IF type='BINARY' THEN DO;
RSUBMIT;
X 'mkdir -p /sub1/sub2/sub3';
PROC UPLOAD INFILE='BINARY.FILE.DSN'
OUTFILE='/sub1/sub2/sub3/binary.file.name'
BINARY STATUS=NO;
RUN;
ENDRSUBMIT;
END;

=====
*** Create a subdirectory & transfer SAS V6+ file ***
=====
IF type='SASV6' THEN DO;
RSUBMIT;
X 'mkdir -p /sub1/sub2/sub3';
PROC UPLOAD INLIB='SAS.FILE.DSN'
OUTLIB='/sub1/sub2/sub3'
STATUS=NO;
RUN;
ENDRSUBMIT;
END;

=====
*** Create a subdirectory & transfer V6 format catalog ***
=====
IF type='SASFORMATV6' THEN DO;
LIBNAME libary 'FORMAT.CATALOG.DSN';
RSUBMIT;
X 'mkdir -p /sub1/sub2/sub3';
LIBNAME newlib '/sub1/sub2/sub3';
OPTIONS NONOTES;
PROC UPLOAD INCAT=LIBRARY._ALL_
OUTCAT=NEWLIB._ALL_ STATUS=NO;
RUN;
ENDRSUBMIT;
END;
=====

```

```
*** Close SAS/CONNECT Session ***
SIGNOFF;
```

As you can see in Figure 3, in batch SAS/CONNECT code the RSUBMIT; and ENDRSUBMIT; commands are used to bracket the SAS code that is to be submitted to the remote machine for execution. In the last part of Figure 3, we show that you can alternate between local execution and remote execution of SAS code by placing code for local execution outside successive blocks of code bracketed by RSUBMIT; and ENDRSUBMIT; statements. This is necessary whenever a transfer requires both local and remote LIBNAME or FILENAME statements.

SAS's X command issues the UNIX mkdir command to create a subdirectory. In this example the -fp flag enables creation of multiple level subdirectories (p) and suppresses any error message that may be issued when requesting creation of a subdirectory that already exists (f). This proved necessary because the upload process would stall if a system error message was generated on the remote UNIX system. These flags may not be exactly the same on other UNIX systems, for example, on SUN Solaris systems just the -p flag seems to accomplish both functions.

When using the X command on a remote system, you will want to include the NOXWAIT option on the statement you use to start the remote SAS session in the SAS/CONNECT script. This prevents the remote system from trying to issue a prompt to return to SAS.

Notice that the PROC UPLOAD command uses three different pairs of input-output keywords and that each pair is related to the type of file being transferred and the conversion performed:

- For EBCDIC file transfer with EBCDIC to ASCII character conversion, use INFILE= and OUTFILE= keywords.
- If you wish a binary image copy of a file, use INFILE= and OUTFILE= keywords with the BINARY option.
- To transfer entire SAS libraries and automatically convert the SAS files to a format appropriate for a UNIX platform, use the INLIB= and OUTLIB= keywords. The advantage of INLIB and OUTLIB is that no MVS LIBNAME statements are needed.

The STATUS=NO option tells PROC UPLOAD not to display the transfer status window. Notice that this method will not work for SAS libraries stored on tape.

- To transfer Version 6 catalogs requires use of the INCAT= and OUTCAT= keywords. These keywords must refer to librefs given on LIBNAME statements -- one for the IBM platform and one for the UNIX platform. Note that the one for the IBM machine is placed before the RSUBMIT; statement so that it executes locally on the MVS system. The _ALL_ keyword used after the librefs saves you from having to worry about the exact names of any catalog entries. SAS/CONNECT will transfer and convert the catalog to the required UNIX format. However, as show in Appendix C, Version 5 format libraries must be converted to Version 6 catalogs prior to transfer.

Figure 4 shows the basic outline for an FTP job step.

Figure 4
Example Code for Batch FTP File Transfer from IBM to UNIX

```
// EXEC PGM=FTP,REGION=4M,PARM=(EXIT)
//SYSPRINT DD SYSOUT=A
//INPUT DD *
some.internet.address
userid
password
locsite autor di
sendsite
binary
put 'IRSS.WORLD.PK.SR04.DICT' /pub/irss/world/pk/sr04.dict
ascii
put 'IRSS.WORLD.PK.SR04.MARG' /pub/irss/world/pk/sr04.marg
ascii
put 'IRSS.WORLD.PK.SR04.SPCRD5' /pub/irss/world/pk/sr04.spcrds
binary
put 'IRSS.WORLD.PY.SR03.DICT' /pub/irss/world/py/sr03.dict
ascii
put 'IRSS.WORLD.PY.SR03.MARG' /pub/irss/world/py/sr03.marg
ascii
put 'IRSS.WORLD.PY.SR03.SPCRD5' /pub/irss/world/py/sr03.spcrds
QUIT
```

In figure 4, we show instream FTP statements following the //INPUT DD * statement so that you can get an idea of how the batch step works. In practice, on the //INPUT DD statement we specified the DSN of the file we created with a SAS program containing FTP statements.

The PARM=(EXIT) parameter on the EXEC statement causes FTP to close the FTP session after the last FTP command. The FTP LOC SITE

command is used for executing commands locally. The AUTOR parameter permits migrated files to be recalled automatically and the DI parameter indicates that each level of a file be treated as a directory. FTP's SENDSITE command prevents MVS from sending information about record format (e.g. data set organization, record length, etc.) which otherwise generates error messages and wastes resources on UNIX systems.

Putting It All Together -- the Program Generator

Now that we've reviewed the basic strategy and some of the important details, let's look at a complete batch job shown in Figure 5. This jobstream assumes that steps one and two outlined above have been completed and all that remains is to process the master list of files created with the TSO DSAT command.

Figure 5
Complete Batch Job for Transferring a Group of Files

```
//PROGGEN JOB UNC.C.P4505.HARDY.K.TIME=5,PTY=9
//PW=X000X
/* Job Step 1 -
/* Allocate files to hold SAS/CONNECT and FTP statements
// EXEC NULL
//ALLOCI DD
DSN=UNCKAH.BIGMOVE.UPLOAD.ROPER.SASCODE,UNIT=SYSDA,
// DCB=(RECFM=FB,LRECL=80,BLKSIZE=6400),
// SPACE=(TRK,(5,2)),DSP=(NEW,CATLG)
//ALLOCI2 DD
DSN=UNCKAH.BIGMOVE.UPLOAD.ROPER.FTPCODE,UNIT=SYSDA,
// DCB=(RECFM=FB,LRECL=80,BLKSIZE=6400),
// SPACE=(TRK,(5,2)),DSP=(NEW,CATLG)
/* Job Step 2 -
/* Execute SAS, to build command files and execute SAS/CONNECT
// EXEC SAS
//SYSIN DD *
*****
*** SAS PROGRAM FOR READING A FILE CREATED BY TSO DSAT ***;
*** COMMAND: DSAT 'RSS.ROPER.' HARDCOPY(dd) and ***;
*** TRANSMITTING ALL FILES EXCEPT DICT FILES & COLUMN BINARY ***;
*** FILES AS TEXT FILES. DICT & CB FILES ARE TRANSFERRED AS ***;
*** BINARY FILES. ***;
*** USES UNCKAH.SAS.CONNECT.CONVEX.BATCH AS CONNECT SCRIPT ***;
***
*** FILL IN THE %LET STATEMENTS BELOW WITH PROPER FILE NAMES ***;
***
*****
*** Input file created by DSAT ***;
*** Create with LRECL=100 ***;
%LET INDSAT=UNCKAH.BIGMOVE.DSATLIST.ROPER.WORK;

*** Allocated above for SAS code *:
*** Output file with LRECL=80 ***;
%LET SASCODE=UNCKAH.BIGMOVE.UPLOAD.ROPER.SASCODE;

*** Allocated above for FTP code *:
%LET FTPCODE=UNCKAH.BIGMOVE.UPLOAD.ROPER.FTPCODE;

*** DSN of SAS/CONNECT Script ***;
%LET CONNECT=UNCKAH.SAS.CONNECT.CONVEX.BATCH;

*****
*****
*** Get DSN's etc. from file created by TSO DSAT command ***;
*****

DATA temp;

LENGTH dsnname $ 60 volume $ 6 ;

FILENAME dsns "&INDSAT" ;
INFILE dsns LRECL=100 PAD ;
INPUT @1 volume $CHAR6. recfm $ 29-31 blksize $ 34-38
       rec1 $ 40-44 @55 dsnname $CHAR45. ;

/* Throw away first line and last line of file */
/* because they don't contain a DSN */
IF volume='SERIAL' OR volume=' ' OR dsnname=' '
  THEN DELETE ;
ELSE OUTPUT;

*****
*** Create text file of SAS commands to be executed ***;
*****

DATA ;
*****
*** Input SAS dataset contain DSN's for files ***;
*** Name of variable for DSN's must be "dsnname" ***;
*****

SET TEMP END=lastname;

*** Maximum length of any dsnname or filename is 60 characters *;
LENGTH tempdsn subdtr restof filename lastdir $ 60;

RETAIN lastdir;

*** Translate DSNNAME into Convex filename and subdtr ***;
*** Assumes that first three levels of DSNNAME will ***;
*** be used for the Convex subdirectory. ***;

*** Find second and third periods in DSN ***;
pos1 = INDEX(dsnname,'.'); tempdsn = SUBSTR(dsnname, pos1+1);
pos2 = INDEX(tempdsn,'.'); tempdsn = SUBSTR(tempdsn, pos2+1);
pos3 = INDEX(tempdsn,'.');
firstdot= pos1 ;
secondot= pos1 + pos2 ;
thirdot = pos1 + pos2 + pos3 ;

*** Translate DSN into lowercase subdtr and filename ***;
*** Adds /pub/irs to front of subdtr ***;
subdtr = '/pub/irs' ||
        LOWCASE(SUBSTR(dsnname,firstdot, thirdot-firstdot));
subdtr = TRIM(TRANSLATE(subdtr, '.', '')) ;
restof = LOWCASE(TRIM(SUBSTR(dsnname, thirdot+1))) ;
filename= TRIM(subdtr) || '/' || TRIM(restof) ;

*****
*** Write out X and FTP Statements for each file type ***;
*****

*****
*** File that contains SAS statements to be %INCLUDED ***;
*****

FILE "&SASCODE" ;

*** Output %IGNON to the Convex statements ***;
IF _N_ = 1 THEN DO ;
  lastdir = " " ; /* Initialize previous subdirectory */
  PUT "%IGNON " ;
  PUT "%SUBMIT " ;
END;

*** If dataset on tape then use SAS/CONNECT ***;
IF volume NE: 'UDAT' AND volume NE: 'MIGRAT' THEN DO ;
IF volume NE: '116618' THEN DO ;
  PUT "*****";
  PUT "X 'mkdir -p " subdtr " " ;
  PUT "PROC UPLOAD INFILE= " dsnname " ;
  PUT " OUTFILE= " filename " STATUS=NO;" ;
  PUT "RUN;" ;
END;

IF volume EQ: '116618' THEN DO ;
```


Figure 6
Batch Code for Transferring an EBCDIC Tape
File from IBM to UNIX

```
*****
*** Create a subdirectory & transfer EBCDIC tape file ***;
*****
FILENAME tape01 'DSN.OF.TAPE.FILE' ; /* open file */
RUN;
RSUBMIT ;
X 'mkdir -fp /sub1/sub2/sub3' ;
PROC UPLOAD INFILE= tape01 /* Use fileref, not DSN */
  OUTFILE= '/sub1/sub2/sub3/file.name'
  STATUS=NO ;
RUN;
ENDRSUBMIT ;
FILENAME tape01 CLEAR ; /*close the file */
```

- 3) Use SAS/CONNECT to transfer RECFM=VB files. We found that when transferring IBM/MVS RECFM=VB data files containing packed decimal data fields and other unique System 370 formats, we needed to use PROC UPLOAD rather than FTP because FTP does not preserve the block length bytes in such files. PROC UPLOAD does preserve these fields (see below for information on reading these files).
- 4) Check transfer speeds for both platforms. We compared transfer rates for SAS/CONNECT and FTP for sessions originating on the MVS system with those originating on the UNIX system and found that the rates for transfers originating on MVS were markedly faster than those originating on the Convex C3480. Your mileage may vary, but check it out.
- 5) Use TBUFSIZE= if you can. According to Horton (1995) the TBUFSIZE= option of the SIGNON statement is the most beneficial one for data transfer. For example,

```
signon tbufsize=32767;
```

increases the amount of data SAS sends to the system's communication buffer to 32k, allowing more data to be passed at a given time. The TBUFSIZE= option can speed transfers on networks using APPC, TCP/IP, DECnet or NetBIOS protocols and has been implemented on the following releases of SAS: V6.08 TS415, V6.09 TS039, V6.10 and V6.11. We were anxious to try this enhancement but unfortunately neither our UNIX release nor our MVS release were updated at the time this would have been convenient.

Tricks for Reading Files on the UNIX System

While we could read and process most text files, SAS files, and SAS catalogs, as we expected on the UNIX system, not all file formats could be read transparently after transfer. Here is a list of some of the tricks we needed for certain file formats.

- 1) EBCDIC text files with RECFM=FB and trailing blanks on records will be translated to a UNIX file with varying length ASCII records when transferred with FTP. This means that you may need to use the LRECL= and PAD options on the SAS INFILE statements or the colon format modifier on the INPUT statement when reading these files. When the same type of file is transferred with SAS/CONNECT, the UNIX file will contain fixed length records, which include any trailing blanks.
- 2) SPSS export format files, can be read with SAS using the SPSS I/O engine or with SPSS using the IMPORT command. See Appendix E for a SAS sample program.
- 3) Data files supplied by the Interuniversity Consortium for Political and Social Research with OSIRIS dictionaries, must have both the data and dictionary file transferred to the UNIX system as binary image files if the data is to be read using the dictionary file with SAS's OSIRIS I/O engine.
- 4) As mentioned in Transfer Tips above, we used PROC UPLOAD to preserve the block length of IBM/MVS RECFM=VB data files containing packed decimal data fields and other unique System 370 formats. We found the resulting UNIX files could easily be read with SAS using the RECFM=S370VB option on the INFILE statement and appropriate EBCDIC and S370 informats on the INPUT statement. Some S370 informats are documented in the V6 SAS Language Reference guide other in SAS Tech Report P-242 pages 16-30.
- 5) UNIX binary image files of IBM/MVS column binary format files must be read with LRECL=160 and RECFM=F on the SAS INFILE statement and appropriate SAS column binary input formats on the INPUT statement.

Making a List and Checking It Twice

Once a group of files had been transferred, we needed to be able to check a list of files on the UNIX system against the master listing on the MVS system. We did this by creating a SAS job on the UNIX platform that recursively listed all files under a top subdirectory stem (e.g. /pub/irss/icpsr), converted the file names back to their IBM/MVS DSN equivalents and saved the results in a SAS file. Then we used SAS/CONNECT to transfer this SAS file back to the MVS platform where we match—merged it with a SAS file containing the DSNs from the original master list created with the TSO DSAT command. Names without matches were simply displayed with PROC PRINT and then resolved. Surprisingly, even when a group of over 7,000 files was transferred and checked, less than a dozen mismatches had to be resolved.

Clearly our method only checked filenames and not file integrity. We placed a tremendous amount of faith in SAS/CONNECT and TCP/IP. However, a sample of files with various formats was read and frequencies of critical variables checked to reassure us somewhat.

Conclusion

We feel our experiences can serve as a prototype for others, especially for other academic sites. We think it is fair to say that base SAS and SAS/CONNECT proved to be invaluable tools in accomplishing this rather large task.

References

Horton, Susan P. (1995) "Maximum Performance Utilization for SAS/CONNECT Processing," Observations, Fourth Quarter 1995, 29-34.

SAS Institute Inc., SAS Language: Reference, Version 6, First Edition, Cary, NC: SAS Institute Inc., 1990

SAS Institute Inc., SAS Technical Report P-242, SAS Software: Changes and Enhancements, Release 6.08, Cary, NC: SAS Institute Inc., 1993

Authors' Addresses

Kenneth A. Hardy
Institute for Research in Social Sciences
20 Manning Hall
CB# 3355
University of North Carolina
Chapel Hill, NC 27599-3355
EMAIL: Ken_Hardy@unc.edu

Arturo Barrios
Office of Information Technology
310 Wilson Library
CB# 3460
University of North Carolina
Chapel Hill, NC 27599-3355
EMAIL: Arturo_Barrios@unc.edu

Sally M. Muller
Office of Information Technology
310 Wilson Library
CB# 3460
University of North Carolina
Chapel Hill, NC 27599-3355
EMAIL: Sally_Muller@unc.edu

Appendix A -- DCB Attributes of Files Found on UNC IBM/MVS System

BMDP

	Disk File Attributes			Tape File
	DATA FILE	PORTABLE FILE	OUTPUT (LISTING FILE)	TAPE FILE
DSORG	PS	PS	PS	PS
LRECL	32768	80	133	DOESN'T MATTER
RECFM	VS	F	UA	DOESN'T MATTER
BLKSIZE	80	80	133	DOESN'T MATTER

SAS--Version 6

	DATA FILE & FORMAT LIBRARY	TAPE FILE & TAPE FORMAT ON DISK
DSORG	PS	PS
LRECL	multiple of 512	32756 (BLKSIZE-4)
RECFM	FS	U
BLKSIZE	same as LRECL	32760 (max permitted for device)

SAS--Version 5.18

	DISK FILE	TAPE FILE	FORMAT LIBRARY
DSORG	DA	PS	PO
LRECL	32756 (BLKSIZE-4)	32756 (BLKSIZE-4)	0
RECFM	U	U	U
BLKSIZE	32760 (max for device)	32760 (max for device)	will vary

SAS--1984

	DISK FILE
DSORG	DA
LRECL	32756 (BLKSIZE-4)
RECFM	U
BLKSIZE	32760 (max permitted for device)

SAS--1982

	DISK FILE ATTRIBUTES	TAPE FILE ATTRIBUTES
DSORG	DA	PS
LRECL	32756 (BLKSIZE-4)	32756 (BLKSIZE-4)
RECFM	U	U
BLKSIZE	32760 (max permitted for device)	32760 (max for device)

SAS--1979

	DISK FILE ATTRIBUTES	TAPE FILE ATTRIBUTES
DSORG	DA	PS
LRECL	32756	32756
RECFM	U	U
BLKSIZE	32760	32760

SPSS

	OLD SPSS (maroon manual, pre SPSS-X)	SYSTEM FILES (Version 1 through 4.1 and SPSS-X)	TRANSPOR T FILES
DSORG	PS	PS	
LRECL	2008/20008	1024	
RECFM	VBS/VB	FB	FB
BLKSIZE	2012	3072 or 4096 (any even multiple of 1024)	3600 (any even multiple of 80)

Appendix B -- Code for Translating Dsnames into UNIX Subdirectory and Filenames

```

*** Input file created by TSO DSAT ***;
*** Allocate with LRECL=1001 ***;
%LET INDSAT=UNCKAH.BIGMOVE.DSATLIST.ICPSR.WORK ;

*****;
*** Get DSN's etc. from file created by TSO DSAT command ***;
*****;

DATA temp;
LENGTH dsn $ 60 volume $ 6 ;
FILENAME dsn "&INDSAT" ;
INFILE dsn LRECL=100 PAD ;
INPUT @1 volume $CHAR6. @55 dsn $CHAR45. ;

/* Throw away first line and last line of file */
/* because they don't contain a DSN */
/* Throw away lines that don't contain a DSN */
IF volume=SERIAL OR volume= OR dsn=

```

```

THEN DELETE;
ELSE OUTPUT;

```

```

*****
*** Translate DSNAMES to subdirectory & file names. ***
*****
DATA :
*****
*** Input SAS dataset containing DSN's for files ***
*** Name of variable for DSN's must be "dsname" ***
*****

SET TEMP END=lastname;

*** Max length of any dsname/filename is 100 chars ***
LENGTH tempdsn subdir restof filename $ 100;

*** Translate DSNAME into Convex filename and subdir ***
*** Assumes that first three levels of DSNAME will ***
*** be used for the Convex subdirectory. ***

*** Find second and third periods in DSN ***
pos1 = INDEX(dsname, '.'); tempdsn = SUBSTR(dsname, pos1+1);
pos2 = INDEX(tempdsn, '.'); tempdsn = SUBSTR(tempdsn, pos2+1);
pos3 = INDEX(tempdsn, '.');
firstdot= pos1;
secondot= pos1 + pos2;
thirdot = pos1 + pos2 + pos3;

*** Translate DSN into lowercase subdir and filename ***
*** Adds /pub/irss to front of subdir ***
subdir = '/pub/irss' ||
LOWCASE(SUBSTR(dsname, firstdot, thirdot-firstdot));
subdir = TRIM(TRANSLATE(subdir, ' ', '.'));
restof = LOWCASE(TRIM(SUBSTR(dsname, thirdot+1)));
filename= TRIM(subdir) || '/' || TRIM(restof);

```

Appendix C -- Batch Code to Move SAS Version 5.18 Format Library from IBM to UNIX

STEP 1: TSO

```

//SASFM518 JOB UNCC.C.,HARDY.K,TIME=1,PTY=9,NOTIFY=UNCKAH
// EXEC SAS.OPTIONS='SASLIB=FRMTLIB'
//FRMTLIB DD DSN=UNCKAH.SAS.SASLIB,DISP=SHR
//TRANFILE DD DSN=UNCKAH.SAS.DATASET.FMTS,DISP=(NEW,CATLG),
// UNIT=DISK,SPACE=(10,5),DCB=(RECFM=FB,LRECL=80,BLKSIZE=8000)
//SYSIN DD *

```

```

LIBNAME TRANFILE XPORT;
PROC VSTOV6 FORMAT=FRMTLIB OUT=WORK;

```

```

PROC FORMAT LIBRARY=WORK CNTLOUT=TRANFILE.FMTS;
RUN;

```

STEP 2: BINARY FTP

STEP 3: CONVEX (note that 'v5fml' below is the name of the transport file on the convex)

```

libname in xport 'v5fml';
libname library 'your-directory';

proc format library=library cntlin=in.fmts fmltblb;
run;

```

Appendix D -- Batch SPSS Program to Convert SPSS System File to SPSS Portable File

```

//SPSSXPT JOB UNCC.C.,HARDY.K,TIME=5,PTY=9
// EXEC SPSS
//IN1 DD DSN=IRSS.ROPER.GSS7291.SPSSX.UNIT=DISK,DISP=SHR
//OUT1 DD DSN=UNCKAH.IRSS.ROPER.GSS7291.SPSSX.XPT,
// DISP=(NEW,CATLG),
// UNIT=DISK,SPACE=(TRK,(500,100),RLSE),
// DCB=(LRECL=80,RECFM=FB,BLKSIZE=9040)
//SYSIN DD *
GET FILE = IN1
EXPORT OUTFILE = OUT1
EXECUTE

```

Appendix E -- Batch SAS Program to Convert Export Format SPSS File to SAS File

```

//CONVERT JOB UNCC.C.,HARDY.K,TIME=1,PTY=9
// EXEC SAS
//SYSIN DD *

LIBNAME IN SPSS 'USERID.SPSS.TEST';

LIBNAME OUT 'USERID.SAS.ONE'
UNIT=DISK DISP=(NEW,CATLG)
SPACE=(TRK,(10,5),RLSE);

DATA OUT.TWO;
SET IN._FIRST;
RUN;

```

SAS, and SAS/CONNECT are registered trademarks or trademarks of SAS Institute Inc. in the USA and other countries. IBM, MVS/TSO, and ISPF are registered trademarks or trademarks of International Business Machines. CONVEX is a registered trademark or trademark of CONVEX Computer Corp. ® indicates USA registration.

Other brand and product names are registered trademarks or trademarks of their respective companies.