Zippen unter z/OS
ZIP-Archiv auf dem Host erzeugen

Frank Hanke
viadee Unternehmensberatung GmbH
Anton-Bruchausen-Straße 8
48147 Münster
Frank.Hanke@viadee.de

Zusammenfassung

Der Beitrag stellt den Umgang mit Zip-Dateien unter SAS 9.2, 9.3 und 9.4 vor und geht dabei auf Unterschiede zwischen den gängigen Betriebssystemen Windows, Linux und z/OS ein. Es wird erläutert, wie Zip-Dateien programmtechnisch mit reinem SAS/Base-Code erstellt werden können, so dass damit beispielsweise auch strukturierte Dateiverzeichnisse samt Inhalt ausgetauscht werden können. Dabei ist es unerheblich, ob die Erstellung oder Verwendung der ZIP-Archive unter z/OS auf dem Mainframe oder unter Windows bzw. Linux erfolgt.
Unter SAS 9.4 ist das Lesen und die Erstellung von Zip-komprimierten Dateien deutlich vereinfacht worden. Die Umstellung auf die Version 9.4 steht jedoch vielen SAS-Installationen, insbesondere in Mainframe-Umgebungen, erst noch bevor. Doch auch unter SAS 9.3 und früheren Versionen können Zip-Archive problemlos über ODS erzeugt oder eingelesen werden; der Programmieraufwand ist etwas höher. In Mainframe-Umgebungen müssen dabei die sogenannten Unix System Services (USS) eingebunden werden, wobei einige Details zu beachten sind.

Schlüsselwörter: Host, Archiv, ZIP, UNIX

1 Einleitung

In vielen DWH-Projekten werden wichtige ETL(Extract, Transform, Load)-Prozesse nach wie vor mit SAS auf einem Großrechner (Mainframe) durchgeführt. Oft werden dann den Fachabteilungen, in einem abgestimmten Prozess die erforderlichen Daten für weitere Datenanalysen und Abfragen regelmäßig auf einem SAS-BI-Server, der unter Windows oder Linux läuft, zur Verfügung gestellt. Wenn dabei große Datenmengen (Anzahl Dateien bzw. Datenvolumina) zwischen den Servern ausgetauscht werden müssen, kann es notwendig sein, die Daten in irgendeiner Form zu komprimieren und einzupacken. Die Vorteile sind, es muss lediglich eine Datei transferiert werden und ganze Verzeichnisbäume können mit vorgegebener Struktur bereitgestellt werden – etwa für ein BI-Portal.

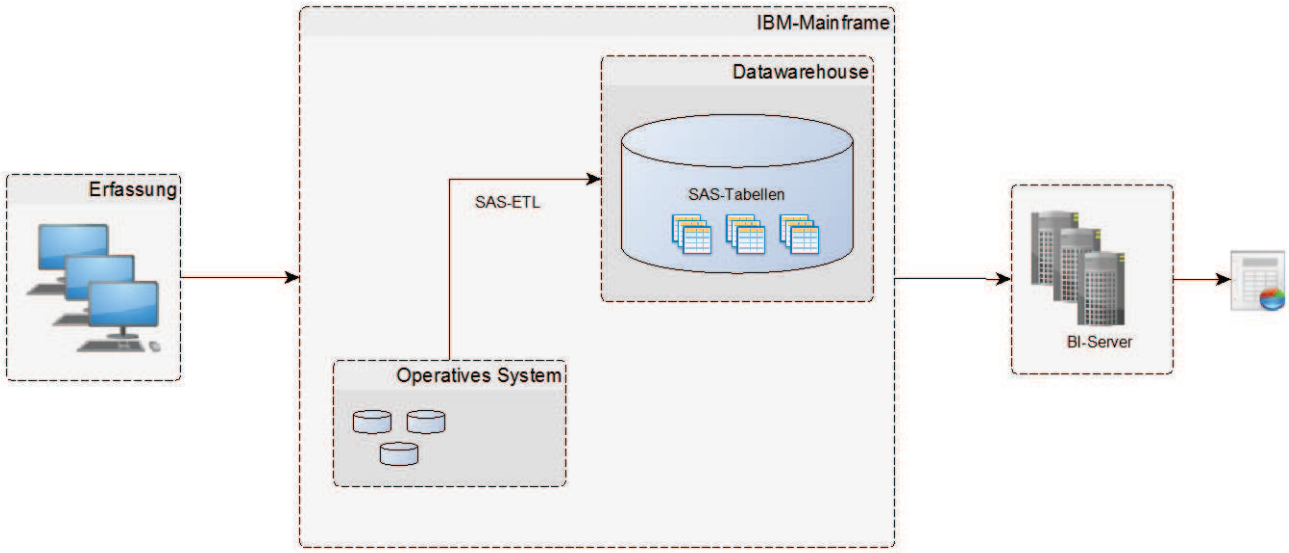

Abbildung 1: Übersichtsbild

2 Zippen mit SAS 9.3 / 9.4

Seit der Version 9.2 können mittels ODS (Output Delivery System) Zip-Dateien erzeugt werden, ohne dass Plattform-spezifische Befehle benutzt werden müssen. Mit dem Release 9.4 hat sich die Erstellung von Archiven weiter vereinfacht. Hier kann die ZIP-Option direkt im Filename-Statement angegeben werden. Die Ausgabe erfolgt direkt als Zip-Datei. Eine Ordnerstruktur kann implizit mit angelegt werden, indem im FILE Statement das Member mit den gewünschten Ordnern angegeben wird.

Tabelle 1: Unterschied SAS 9.3 - SAS 9.4

SAS 9.3	SAS 9.4

`filename sas93 'd:\temp\shoes.txt';`
`data _null_;`
`  file sas93;`
`  set sashelp.shoes;`
`  put region $25. product $14.;`
`run;`
`ods package (sas93Output)`
`  open nopf;`
`ods package (sas93Output)`
`  add file=sas93;`
`ods package (sas93Output)`
`  publish archive`
`  properties(archive_name='sas93.zip'`
 	
`archive_path='d:\temp');`
`ods package (sas93Output) close;`
	`filename foo zip`
	`'d:\temp\sas94.zip';`
	`data _null_;`
	`  file foo(shoes.txt);`
	`  set sashelp.shoes;`
	`  put region $25. product`
	`$14.;`
	`run;`

Mit dem Release 9.2 oder 9.3 sind einige zusätzliche Schritte notwendig. Zunächst wird die zu zippende Datei temporär als Datei erzeugt. Anschließend wird ein ODS-Package, in diesem Fall mit dem Namen sas93Output, geöffnet. Die Option „NOPF“ bewirkt, dass kein zusätzlicher Eintrag mit Metadaten in der Zip-Datei erstellt wird. Nachdem alle Dateien über das „add-Statement“ hinzugefügt wurden, wird das Archiv mittels „publish Archive“ erzeugt. Beim Add-Statement kann mit dem Zusatz Path= auch ein gewünschter relativer Pfad innerhalb der Zip-Datei erzeugt werden. In diesem Beispiel wird die Datei sas93.zip im Verzeichnis d:\temp erstellt. Am Ende muss das ODS-Package noch geschlossen werden.

Neu mit Version SAS 9.4 ist, dass mittels des FILENAME Statements auch ZIP Archive wieder eingelesen werden können, ohne diese vorher auspacken zu müssen. Hierzu kann die Member-Erweiterung des FILENAME Statements genutzt werden, um die FILEREF direkt auf das spezielle, einzulesende Member des Archivs zu lenken.

3 Host Exkurs

Dieses Kapitel soll und kann nur einen kleinen Einblick in die Welt des Mainframes geben. Interessant jedoch ist, dass Mainframes heute noch gebaut werden, obwohl immer wieder behauptet wird, der Host sei tot. IBM hat erst im Januar 2015 eine neue Mainframe-Generation z13 vorgestellt.

Supercomputer werden i. d. R. auf hohe Rechenleistung hin entwickelt. Im Gegensatz dazu sind Großrechner eher auf Zuverlässigkeit und hohen Datendurchsatz (I/O) ausgelegt. Die typischen Anwendungen eines Großrechners sind hochzuverlässige Verarbeitung von Massendaten, beispielsweise bei Banken, Versicherungen, großen Unternehmen und in der öffentlichen Verwaltung.

Der Zugriff auf den Großrechner erfolgt i. d. R. über eine Terminalemulation am PC.

3.1 Anmeldung

Nach der Anmeldung wird je nach Konfiguration des Mainframes ein Eingangspanel angezeigt. Von hier kann der Benutzer in weitere Unterpanels verzweigen und unterschiedliche Aufgaben erledigen. Das Layout der Panels kann von Konfiguration zu Konfiguration abweichen; die grundlegende Arbeitsweise bleibt jedoch immer gleich. Die Optionen können verkettet mit einem Punkt eingegeben werden.

- Job Queue anzeigen => s.st
- Dataset anlegen => 3.2
- Datasets anzeigen => 3.4

Neben den aufgelisteten Optionen (0, 1 … X), können beispielsweise über den Befehl „start“ bis zu 9 Bildschirme geöffnet werden.Mit dem „=“ Operator, können Befehle vom Eingangspanel von überall gestartet werden. Beispiel: =s.st Anzeige der Job Queue.

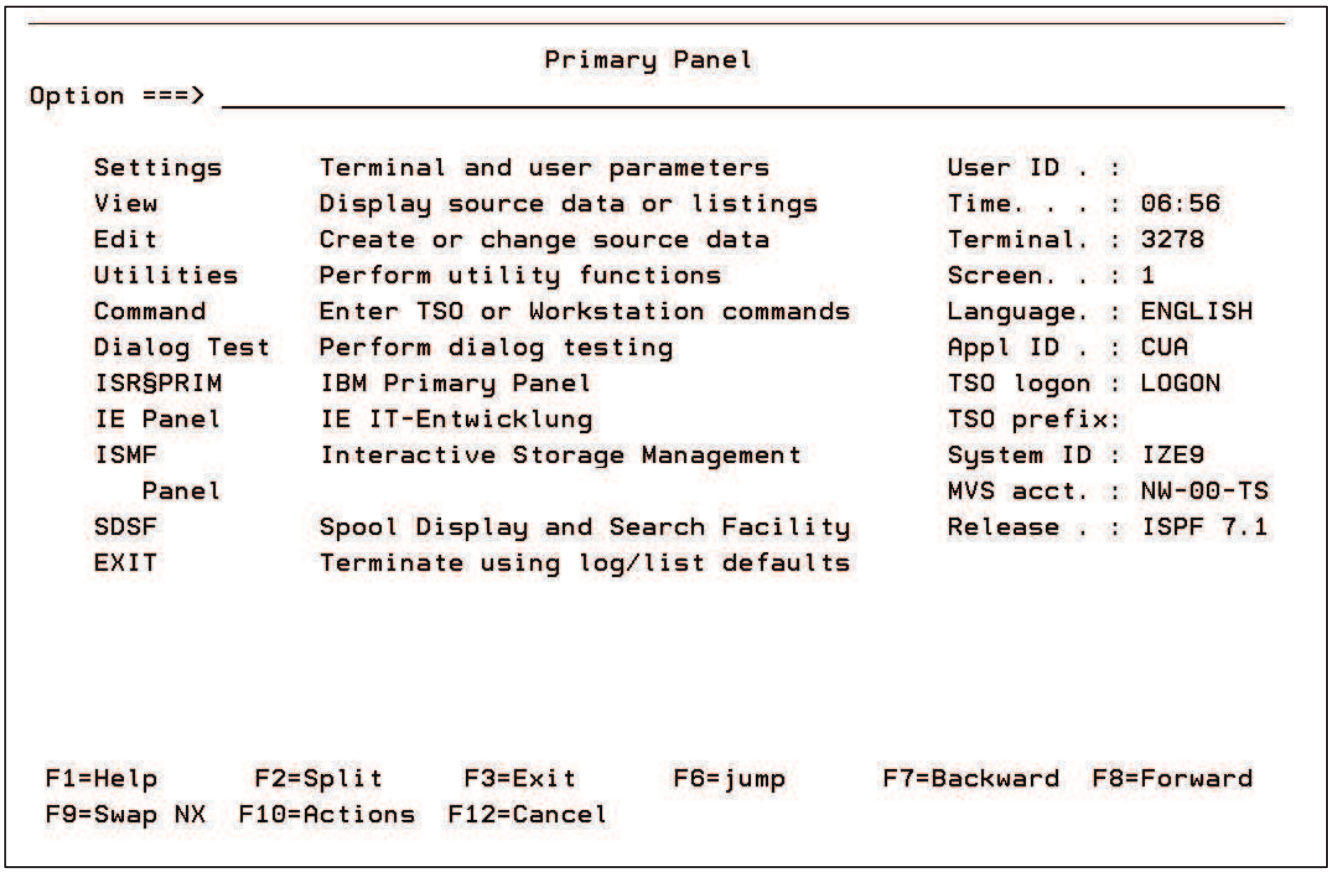

Abbildung 2: Einstiegspanel

3.2 Verzeichnisse / Dateien

Verzeichnisse bzw. Dateien werden im Hostumfeld als Datasets bezeichnet. Jedes Dataset kann direkt eine Datei sein (analog zu einer dedizierten Datei auf einem Windows-System) oder viele Dateien enthalten (analog eines Verzeichnisses unter Windows). Das hängt von der Organisation des Datasets ab.
Die Namen von Datasets setzen sich aus sogenannten Qualifiern zusammen, die nicht länger als 8 Zeichen sein dürfen und mit Punkten voneinander getrennt sind. Die Gesamtlänge eines Dataset-Namens ist auf 44 Zeichen, inklusive der Punkte, beschränkt.

Beispiel: DE.VIADEE.FRANK

Wenn Datasets oder Dateien angelegt werden, muss im Vorfeld klar sein, wie viel Speicher mindestens benötigt wird und wie sich das Dataset verhalten soll, wenn der ursprünglich vorgesehene Speicherplatz erschöpft ist. Bis zu einer gewissen Größe (i.d.R. 16 Extents) kann ein Dataset automatisch vergrößert werden; andernfalls muss es neu angelegt und der Inhalt kopiert werden.

3.3 Programme starten

Programme auf dem Mainframe können nicht einfach per Auswahl gestartet werden. Soll ein Programm auf dem Host im Batch-Betrieb laufen, so muss um den Programmaufruf ein Rahmen in der sogenannten JCL (Job Control Language) geschrieben wer-

den. Die JCL ist die Steuersprache für Stapelverarbeitungen im Großrechnerumfeld. Über den Befehl „sub“ wird der Job vom JES (Job Entry Subsystem) eingelesen und interpretiert. Der Benutzer kann das Logfile in der Job Queue (zur Erinnerung der Befehl =s.st) einsehen.

Für Programme (SAS, Cobol) ist die Zeilenlänge auf 80 Zeichen beschränkt.

Beispiel-JCL eines Programmaufrufs:

```
//DM#KSAGX JOB (XX-00-TSO0-0000),'HANKE FRANK',
//         MSGCLASS=H,CLASS=L,REGION=256M,NOTIFY=&SYSUID
//*-------------------------------------------------------------+
//PROCS   JCLLIB ORDER=(SASPROC.SP3B.SAS.SRC.PROC)
//STEP100 EXEC SAS,
//          DB2INST=DB2TST,
//          MANDANT=XX
//*************************************************************/
//* SAS-PROGRAMM EINBINDEN
//*************************************************************/
//SYSIN    DD *
%include PRGCNT($DM$KSAG);

//*************************************************************/
//* DATEN-LIBRARYS WERDEN ALLOKIERT
//*************************************************************/
//TMPS01   DD DSN=VIADEE.SP3B.SASTST.LIB.TMPS01,DISP=OLD
//DATKSAA  DD DSN=VIADEE.SP3B.SASTST.LIB.MA.DATKSA,DISP=OLD
//DATKSAR  DD DSN=VIADEE.SP3B.SASTST.LIB.MR.DATKSA,DISP=OLD
//PARAM    DD DSN=VIADEE.SP3B.SASTST.SRC.PRGSYSIN(DM$KSAG),DISP=SHR
//PRGSYSIN DD DSN=VIADEE.SP3B.SASTST.SRC.PRGSYSIN(GM),DISP=SHR
```

3.4 Unix-Shell unter z/OS

Auf Mainframes stehen unter z/OS Unix-Dienste bereit, die sogenannten Unix System Services (USS). Diese Unix-Implementierung ermöglicht es, Unix-Anwendungen auf IBM-Großrechnern laufen zu lassen und UNIX basierte Dienste zu realisieren.

Beispielsweise stehen zwei Shells Verfügung:

- z/OS-Shell
- tcsh-Shell

Die z/OS-Shell lehnt sich an die Unix-System V-Shell an und besitzt einige Merkmale der Korn-Shell (ksh). Die tcsh-Shell ist eine erweiterte aber kompatible Version der Berkley Unix C-Shell (csh). Als Standard ist im RACF (Resource Access Control Facilitiy, Berechtigungssystem auf dem Host) die z/OS-Shell eingestellt. Mit dem Befehl `LISTUSER <Username> OMVS` können die Einstellungen angezeigt werden.

Beispiel:

```
UID=4711
HOME=/u/user
PROGRAM=/bin/sh
```

Mit dem Befehl `ALTUSER <USERNAME> OMVS(PROGRAM='/bin/tcsh'))` kann der Standard überschrieben werden.

Der Zugriff auf die Unix-Shell erfolgt über den Befehl `TSO OMVS`. Weitere Zugriffsmöglichkeiten bestehen mittels rLogin und telnet.

```
IBM
Licensed Material - Property of IBM
5650-ZOS Copyright IBM Corp. 1993, 2013
(C) Copyright Mortice Kern Systems, Inc., 1985, 1996.
(C) Copyright Software Development Group, University of Waterloo, 1989.

U.S. Government Users Restricted Rights -
Use,duplication or disclosure restricted by
GSA ADP Schedule Contract with IBM Corp.

IBM is a registered trademark of the IBM Corp.

FSUM2386 No shell program was specified in the user profile.  The default shell
('/bin/sh') is used.
FSUM2383 No initial directory pathname was specified in the user profile.  The h
ome directory is set to root.
$



 ===>
                                                                        INPUT
ESC=Ä    1=Help      2=SubCmd    3=HlpRetrn  4=Top       5=Bottom    6=TSO
         7=BackScr   8=Scroll    9=NextSess 10=Refresh  11=FwdRetr  12=Retrieve
```

Abbildung 3: Unix-Shell

3.5 SAS Interaktiv

Ähnlich wie unter Windows oder Linux, ist es auch auf dem Mainframe möglich, eine interaktive SAS-Session zu starten. Voraussetzung ist, der Benutzer hat die entsprechenden Berechtigungen und SAS ist korrekt konfiguriert. Im unteren Teil, dem Program-Editor, kann der Benutzer SAS Code eingeben und über den Befehl „sub“ ausführen.

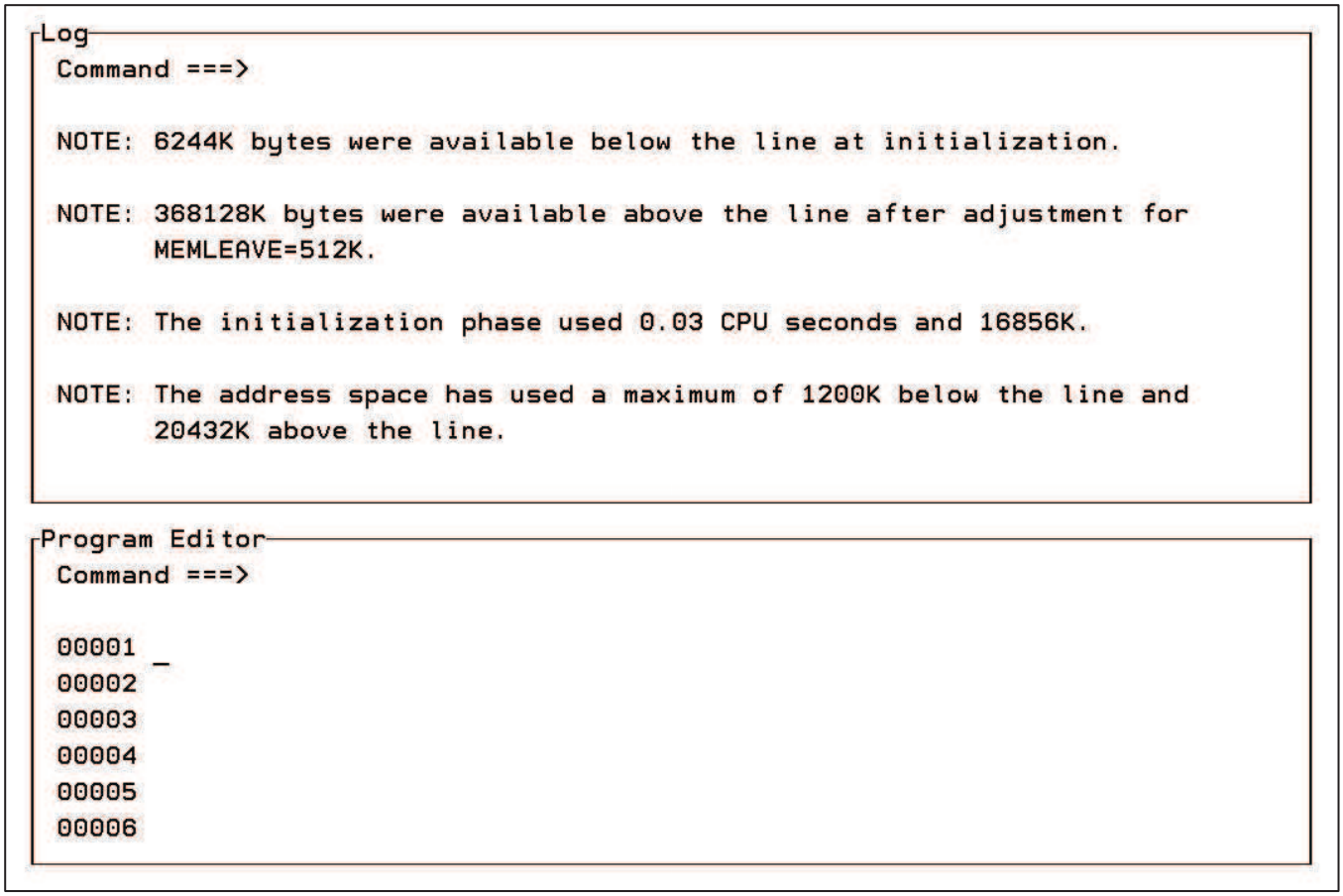

Abbildung 4: Interaktive SAS-Session

Output PROC PRINT suspended
Command ===>

The SAS System 1
07:15 Friday, March 18, 2016

Obs	Name	Sex	Age	Height	Weight
1	Alfred	M	14	69.0	112.5
2	Alice	F	13	56.5	84.0
3	Barbara	F	13	65.3	98.0
4	Carol	F	14	62.8	102.5
5	Henry	M	14	63.5	102.5
6	James	M	12	57.3	83.0
7	Jane	F	12	59.8	84.5
8	Janet	F	15	62.5	112.5
9	Jeffrey	M	13	62.5	84.0
10	John	M	12	59.0	99.5
11	Joyce	F	11	51.3	50.5
12	Judy	F	14	64.3	90.0
13	Louise	F	12	56.3	77.0
14	Mary	F	15	66.5	112.0
15	Philip	M	16	72.0	150.0

R

Abbildung 5: Ausführung eines Proc Print auf sashelp.class

```
Program Editor
Command ===>

00001 proc print data=sashelp.class;run;_
00002
00003
00004
00005
00006
```

Abbildung 6: Logfile

4 Unix Befehle in einer SAS-Session ausführen

In einer SAS-Session gibt es unterschiedliche Möglichkeiten, Unix-Kommandos oder andere Systembefehle auszuführen. Die Unix-Kommandos funktionieren wegen des Vorhandenseins der USS dabei auch unter z/OS. In diesem Kapitel sollen zwei Möglichkeiten kurz vorgestellt werden. Im Übrigen funktioniert der Mechanismus auch mit Dos-Befehlen, wie „Dir“.

4.1 X

Mit dem Kommando X wird ein Systembefehl gesendet und eine Shell geöffnet, in der dieser abläuft. Auf dem Host kann jedoch keine Shell geöffnet werden, so dass der Anwender kein Feedback bekommt, ob der Befehl ausgeführt wurde.

4.1.1 Syntax

X Unix-Kommando;
Auf dem Host zum Unix-System => X ls –ltr;
Unter Windows => X Dir;

4.2 Pipe

Über eine Pipe können die Ausgaben von Systembefehlen an das SAS-Programm zurückgeliefert werden. Ausgelöst wird der Aufruf durch das „infile“ im Datastep. Soll vom Host ein Unix-Befehl über eine Pipe ausgeführt werden, muss zwingend die Filesystem-Option vorher auf HFS eingestellt werden. Ansonsten würde der Befehl an den Host gesendet, der diesen nicht interpretieren kann.

4.2.1 Beispiel anhand eines Dos-Befehls

Der Aufruf eines Unix-Befehls auf dem Host funktioniert analog.

```
* Definition des aufzurufenden Befehl
filename dir pipe "Dir";

* Mit X das Verzeichnis wechseln
X cd C:\KSFE-2016;

data dirlist;
     infile dir;
     input  Datum   : $10.
            Uhrzeit : $10.
            Typ     : $80.
            Name    : $80.;
run;
```

Tabelle 2: Ausgabe des Befehls "Dir"

Datum	Uhrzeit	Typ	Name
Datentr„ger	in	Laufwerk	C:
Volumeseriennummer:	8C2E-F178	Verzeichnis	von
25.01.2016	17:33		
25.01.2016	17:33		..
21.01.2016	18:00		Grafiken
18.01.2016	17:55		JCL
21.01.2016	17:57	6.377	KSFE-2016.mm

Datum	Uhrzeit	Typ	Name
13.01.2016	06:49	2.951.985	KSFE2016.zip
08.01.2016	10:47	222.314	SAS
23.01.2016	11:16		Screenshots
16.01.2016	15:35	21.415	ZIP.egp
11.01.2016	06:43	1.131.640	Zippen
23.01.2016	11:23	2.766.276	Zippen
10	Datei(en),	7.657.108	Bytes
6	Verzeichni	11.808.448.512	Bytes

5 Zippen unter z/OS mit SAS

Unter dem Betriebssystem z/OS wird als Ablage von Zip-Dateien nur das Filesystem unter Unix (hierarchical file system, HFS) unterstützt. Im Dateisystem vom Host (MVS, Multiple virtual Storage) ist das Erstellen von Zip-Dateien nicht möglich.

Um aus SAS heraus auf das Unix-Dateisystem greifen zu können, stehen vier Möglichkeiten zur Verfügung:

- Einen Slash oder eine Tilde im Pfadnamen voranstellen
  ```
  filename datei1 ´/u/tmp/datei1.txt´
  filename datei2 ´~/tmp/datei2.txt´
  ```
- HFS als Dateityp angeben
  ```
  filename datei3 hfs ´datei3.txt´
  ```
- HFS als Datei-Präfix verwenden
  ```
  filename datei4 ´HFS:datei4.txt´
  ```
- **Filesystem-Option setzen**
  ```
  options Filesystem=HFS
  filename datei5 ´/tmp/datei5.txt´
  ```

  Die Option bleibt während der SAS-Session erhalten. Sofern im weiteren Ablauf Daten auf dem Host geschrieben werden, ist es wichtig, diese Option nach dem Zugriff auf das Unix-System wieder auf den Wert MVS zu setzen.

In dem hier vorgestellten Beispiel wird die Variante mit der Filesystem-Option genutzt.

5.1 Ablauf

Aus der fachlichen Anforderung ergibt sich folgende Vorgehensweise. Auf die einzelnen Schritte wird in den nächsten Unterkapiteln eingegangen.

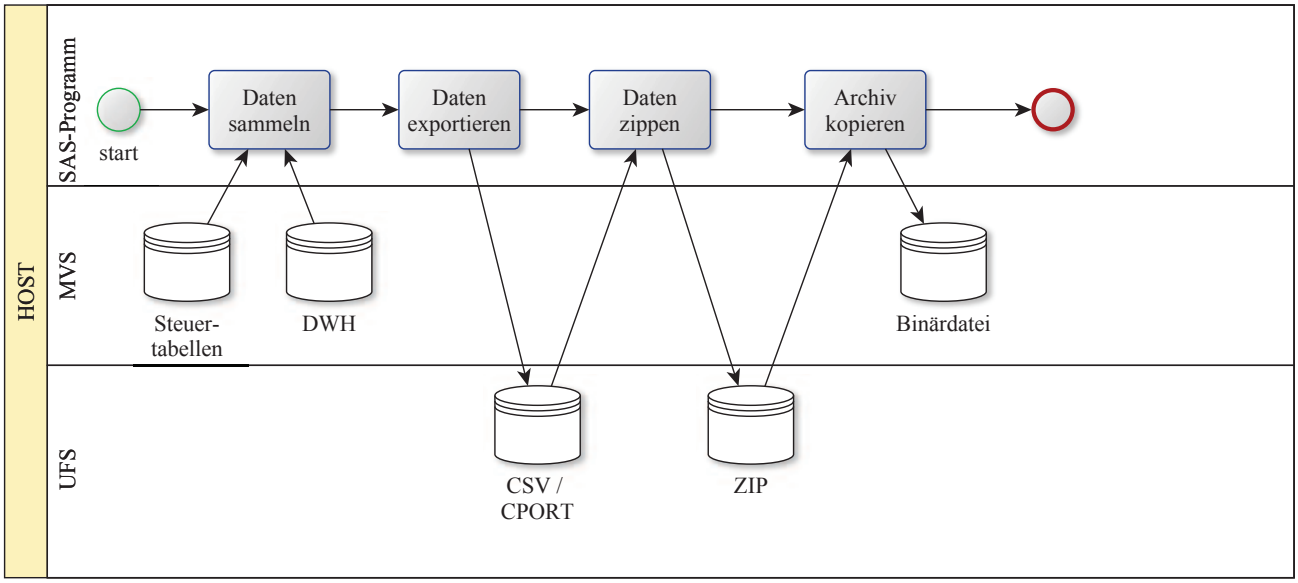

Abbildung 7: Schematischer Ablauf

5.2 Daten Sammeln und Exportieren

Zunächst müssen allen Daten / Tabellen zusammengestellt werden, die vom Anwender in der Zip-Datei angefordert wurden. In Steuertabellen ist definiert, welche Daten wie zur Verfügung gestellt werden sollen. Die Daten werden direkt in das Unix-Filesystem geschrieben.

5.2.1 Export als CPORT

Mit dieser Option können gesamte Bibliotheken oder einzelne Tabellen aus einer Bibliothek exportiert werden. Weitere Einschränkungen sind nicht möglich.

5.2.2 Export als CSV

Mehr Möglichkeiten bietet der Export als CSV-Datei. Hier können in Steuertabellen einzelne Filter definiert, Spalten ausgeschlossen bzw. umbenannt und Formate geändert werden, beispielsweise bei täglichen Lieferungen, mit der Einschränkung auf ein bestimmtes Tagesdatum.

5.2.3 Beispiel

In diesem Beispiel sollen zwei Tabellen, einmal als CSV- und einmal als CPORT-Datei, exportiert werden. Nach diesem Schritt sind im Unix-Dateisystem des Mainframes im Verzeichnis /tmp die beiden Dateien zu finden.

```
* Filesystem umschalten;
OPTIONS FILESYSTEM=HFS;

* Dateireferenzen definieren;
Filename unixcsv '/tmp/sashelp.csv' RECFM=V LRECL=15000;
```

```
Filename unixcpr '/tmp/sashelp.cpr' RECFM=F LRECL=80;

* als csv Exportieren;
Data _Null_;
  Set Sashelp.Class;
  File unixcsv dsd dlm=";";
  Put (_all_)(~);      * => Tipps & Tricks KSFE-2015 [1]
Run;

* als cport exportieren;
proc cport file=unixcpr lib=Sashelp;
    select Class;
run;
```

5.3 Zip-Archiv erstellen

In diesem Schritt werden die gesammelten Daten zu einem Dateiarchiv zusammengepackt. Das Ergebnis wird, wie die beiden anderen Dateien, direkt im Verzeichnis /tmp unter Unix erstellt. Weil aktuell SAS 9.3 als Umgebung installiert ist, erfolgt die Erstellung über das Output Delivery System.

5.3.1 Beispiel

```
* zip-Datei erzeugen;
ODS PACKAGE OPEN NOPF;
ODS PACKAGE ADD File=unixcsv
                TEXT
                Path='/tmp/';

ODS PACKAGE ADD File=unixcpr
                BINARY
                Path='/tmp/';

ODS PACKAGE PUBLISH ARCHIVE PROPERTIES
          (ARCHIVE_NAME="sashelp.zip"
           ARCHIVE_PATH="/tmp/");

ODS PACKAGE CLOSE;
```

```
-rw-r--r--   1          SYS1          925 Mar 18 07:28 sashelp.zip
-rw-r--r--   1          SYS1         1369 Mar 18 07:28 sashelp.cp
$
 ===>
                                                                INPUT
ESC=Ä   1=Help      2=SubCmd    3=HlpRetrn  4=Top       5=Bottom    6=TSO
        7=BackScr   8=Scroll    9=NextSess 10=Refresh  11=FwdRetr  12=Retrieve
```

Abbildung 8: Dateiliste nach dem Zippen

5.4 Transfer und Housekeeping

Als letzter Schritt muss die erzeugte ZIP-Datei binär zurück auf den Host transferiert werden, damit die Daten letztendlich dem Benutzer zur Verfügung gestellt werden können. Am Ende werden die Dateien im Unix-System entfernt. Der Unix-Befehl „rm“ wird über den Pipe-Mechanismus ausgeführt.

5.4.1 Beispiel

In diesem Fall wird der Transfer zwischen dem Host und dem Unix-System über ein Makro realisiert, das die Datei binär (byteweise) in ein Dataset kopiert. Die Zieldatei, also der logischen Name auf dem Host, ist in der JCL allokiert und muss daher hier nicht als Filename aufgeführt werden. Die Hilfsmakros M_CPBIN und M_UNIX werden weiter unten erläutert.

```
* zip-Datei zum Host binaer transferieren
OPTIONS FILESYSTEM=HFS;
FILENAME quelle "/tmp/sashelp.zip";
OPTIONS FILESYSTEM = MVS;
* Zieldatei ist in der JCL Allokiert s. u.

%M_CPBIN( QUELLREF  = quelle,
          ZIELREF   = ZIPDAT,
          FEHLERVAR = rc);

* Dateien entfernen
%M_UNIX(BEFEHL = rm &quelle.);
* zum Schluss das Host-Dateisystem einstellen
OPTIONS FILESYSTEM = MVS;
```

5.4.2 Makro M_CPBIN

Parameter:

```
QUELLREF  =
ZIELREF   =
RECSIZE   = 8196
FEHLERVAR =
```

`Quellref` wird über ein Filename-Statement erzeugt und beim Aufruf als Parameter übergeben. Sollen Daten von z/OS-Unix kopiert werden, muss vor dem Filename-Statement die `Option Filesystem=HFS` gesetzt werden. Die Blockgröße, also wie viele Bytes auf einmal gelesen werden, ist mit 8192 Bytes vorbelegt. In der Variable `FEHLERVAR` wird der letzte Fehlercode zurückgeliefert, der im Folgeprogramm ausgewertet werden kann.

```
    DATA _NULL_;
        * Variable, die einen Block speichert *;
        LENGTH record $ &RECSIZE.;
        * Variable für das Ausgabeformat eines Blocks *;
        LENGTH ausgabeformat $32;
```

```
        * Variablen für die Datei-IDs *;
        ATTRIB eing_id LENGTH=8
               ausg_id LENGTH=8;
        * Für Fehlerprüfungen: *;
        alles_ok = 0; * initial TRUE *;

        eing_id = FOPEN("&QUELLREF.", "S", &RECSIZE., "B");

        IF eing_id NE 0 THEN DO;

            ausg_id = FOPEN("&ZIELREF.", "O", &RECSIZE., "B");

            IF ausg_id NE 0 THEN DO;
                * Solange noch weitere Bytes in der Datei sind: *;
                DO WHILE (FREAD(eing_id)=0);
                    call missing(ausgabeformat, record);
                    * Einen Record einlesen: *;
                    rc = FGET(eing_id, record, &RECSIZE.);
                    * Position wird benötigt für Schreibvorgang: *;
                    columnPos = fcol(eing_id);
                    IF (columnPos - &RECSIZE.) = 1
                    THEN auslaenge = &RECSIZE.;
                    ELSE auslaenge = columnPos - 1;
                    * Fehler merken: *;
                    IF rc ne 0 THEN alles_ok = 1;
                    * Record rausschreiben, wieder Fehler merken: *;
                    ausgabeformat = cats("$char", auslaenge, ".");
                    rc = FPUT(ausg_id, PUTC(record, ausgabeformat));
                    IF rc ne 0 THEN alles_ok = 2;
                    rc = FWRITE(ausg_id);
                    IF rc ne 0 THEN alles_ok = 3;
                END;
            END;
            * Datei schließen *;
            rc = FCLOSE(ausg_id);
        END;
        rc = FCLOSE(eing_id);
    RUN;
```

Die Eingabedatei wird im Binärformat zum sequentiellen Einlesen geöffnet. Analog dazu wird die Ausgabedatei zum Schreiben geöffnet. Über eine Schleife werden die Daten blockweise, in diesem Fall mit einer Satzlänge von 8192 Bytes, gelesen und in Blöcken hintereinander in die Ausgabedatei geschrieben. Die zu schreibende Satzlänge wird mittels der aktuellen, relativen Cursorposition im File Data Buffer (FDB) ermittelt, da der letzte Block durchaus kürzer als die definierte Satzlänge sein kann. Über die `cats`-Anweisung wird das aktuelle Ausgabeformat festgelegt: „`$char8192.`“.

5.4.3 Makro M_UNIX

```
%put Makro M_UNIX führt den Befehl;
%put &BEFEHL.;
%put aus.;

%local datei;
%let datei=datei;

%let rc = %sysfunc(filename(datei, &befehl., pipe));
%IF &rc. EQ 0
%THEN %DO;

     data work.unix_befehlsausgabe;
         infile datei truncover;
         length ausg $ 200;
         input ausg $ char200.;
     run;

     TITLE "Makro M_UNIX: Ausgabe zu Unix-Befehl
                 %BQUOTE(&befehl.)";
     proc print data = work.unix_befehlsausgabe;run;
     TITLE;
 %END;
 %ELSE DO;
    %PUT Makro M_UNIX: Fehler bei der Ausführung des Befehls!;
 %END;
```

Als erstes wird im SAS-Log der auszuführende Befehl ausgegeben. Über %sysfunc wird die Funktion „Filename“ aufgerufen, die ein entsprechendes Statement erzeugt, in diesem Fall eine Pipe. Die Definition der Makrovariablen ist notwendig, damit die Filereferenz „datei“ im weiteren Verlauf bekannt ist.

```
Filename datei pipe „rm /tmp/sashelp.zip“;
```

Der Unix-Befehl wird über die Anweisung „`infile Datei`“ im Datastep ausgelöst. Die Rückgabewerte werden zeilenweise über das „Input-Statement“ eingelesen und in der SAS-Tabelle „work.unix_befehlsausgabe“ gesichert.

Anschließend wird der Inhalt der SAS-Tabelle mittels „proc print“ ausgegeben. Wenn der Aufruf der Filename-Funktion fehlgeschlagen ist, wird eine entsprechende Meldung in das Logfile geschrieben.

6 Fazit

Für regelmäßig abgestimmte Lieferungen von vielen Daten in eine Datei, die vom Host zum Anwender transportiert werden muss, kann das Zippen eine echte Alternative sein. Das Zusammenspiel zwischen z/OS und dem Unix-Dateisystem ist mit SAS einfach zu realisieren.
Für die Erstellung von Archiven ist keine zusätzliche Software von Drittanbietern notwendig. SAS bringt von Haus aus alle notwendigen Mittel und Werkzeuge mit.
Der Entwicklungsaufwand ist anfangs etwas höher, da die Daten vom Unix-System wieder zurück zum Host kopiert werden müssen.

Literatur

[1] Tipps & Tricks KSFE-Tagung 2015, Hannover