-

z/OS Job Control Language

Musterlösungen zu den Übungen

18. März 2010

Eine Ausarbeitung von:

cps4it
Ralf Seidler • Stromberger Straße 36A • 55411 Bingen
Fon: 06721-992611 • Fax: 06721-992613 • Mail: ralf.seidler@cps4it.de
Internet : http://www.cps4it.de
Steuernummer: 08/220/2497/3, Finanzamt Bingen, Ust-ID : DE214792185

Diese Seite bleibt frei

Inhaltsverzeichnis

1 Vorbereitungen

Zum Schmunzeln

"Glück ist: zu begreifen, wie alles zusammenhängt."

Sten Nadolny (*1942), dt. Schriftsteller

1.1 Anmelden und Test der User-Iden

Melden Sie sich nach Vorgabe im TSO an.
Melden Sie sich wieder ab und erneut wieder an.
Konfigurieren Sie Ihre TSO-Session, wie Sie am besten arbeiten können.
Testen Sie, ob Sie auf die Dateien des Referenten lesend zugreifen können.

Vorgehensweise

wie erklärt / bekannt

1.2 Bibliothek für Übungsaufgaben erstellen

Erstellen Sie eine PO-Datei, um die Übungsaufgaben dieses Seminars aufnehmen zu können.

Name der Bibliothek	userid.KURS.JCL
Satzlänge	80 Byte
Satzformat	fest geblockt
Dateigröße	5 Spuren primär, 2 Spuren sekundär
Directory	Platz für 30 Member

Kontrollieren Sie das Ergebnis.

Vorgehensweise

Definition über ISPF Menü 3.2.A (Allocate new Dataset)

```
  Menu  RefList  Utilities  Help
------------------------------------------------------------------------------
--
                            Allocate New Data Set
Command ===>
                                                                        More:
+
Data Set Name  . . . : RZSR.KURS.JCL

Management class . . .                 (Blank for default management class)
Storage class  . . . .                 (Blank for default storage class)
 Volume serial . . . .                 (Blank for system default volume) **
 Device type . . . . .                 (Generic unit or device address) **
Data class . . . . . .                 (Blank for default data class)
 Space units . . . . . TRACK           (BLKS, TRKS, CYLS, KB, MB, BYTES
                                        or RECORDS)
 Average record unit                   (M, K, or U)
 Primary quantity  . . 5               (In above units)
 Secondary quantity    2               (In above units)
 Directory blocks  . . 8               (Zero for sequential data set) *
 Record format . . . . FB
 Record length . . . . 80
 Block size  . . . . .
 Data set name type  :                 (LIBRARY, HFS, PDS, or blank)  *
                                       (YY/MM/DD, YYYY/MM/DD
 Expiration date . . .                  YY.DDD, YYYY.DDD in Julian form
```

2 Jobkarte

2.1 Erstellen einer Jobanweisung

Member: $JOBCARD

Kodieren Sie eine JOB Anweisung, eine Jobkarte, gemäß den Vorgaben in der Firma.
Welche Angaben sind verpflichtend?
Schreiben Sie eine minimale JOB Anweisung.
Welche Angaben sind frei wählbar?
Welche Angaben sind freiwillig?
Lassen Sie den Job laufen. Was geschieht?

Vorgehensweise

Editieren userid.KURS.JCL($JOBCARD) mit Inhalt wie unten angegeben.
Die Accountnummer entweder von Kollegen kopieren oder auf TSO-Einstiegsseite kopieren (dort wo User-ID und Passwort einzugeben sind).

```
//RZSRGEN JOB (000947,TEST,,,,9900),'ZIT P 2.21,SEIDLER'
```

Ergebnis: Der Job ist zwar startbar, aber das Ergebnis ist – je nach Firma – nirgendwo sichtbar.

Hinweis: Die Accountnummer ist abhängig von der Firmenvorgabe.

2.2 Jobkarte erweitern - 1

Member: $JOBCARD

Erweitern Sie die Jobkarte mit dem Parameter CLASS.
Erweitern Sie die Jobkarte mit dem Parameter MSGCLASS.
Lassen Sie den Job laufen und analysieren Sie die Ausgabe.

Vorgehensweise

Editieren userid.KURS.JCL($JOBCARD) mit Inhalt wie unten angegeben

```
//RZSRGEN JOB (000947,TEST,,,,9900),'ZIT P2.21,SEIDLER',
//            MSGCLASS=T,CLASS=G
```

Hinweis: MSGCLASS ist abhängig von Firma (z.B. Commerzbank: T, R+V: Y)

2.3 Jobkarte erweitern - 2

Member: $JOBCAR2

Erweitern Sie die Jobkarte mit dem Parameter NOTIFY.
Lassen Sie den Job mit unterschiedlichen Angaben für NOTIFY laufen und analysieren Sie die Ausgabe.

Erweitern Sie die Jobkarte mit dem Parameter REGION.
Lassen Sie den Job mit unterschiedlichen Angaben für REGION laufen und analysieren Sie die Ausgabe.

Erweitern Sie die Jobkarte mit dem Parameter TIME.
Lassen Sie den Job mit unterschiedlichen Angaben für TIME laufen und analysieren Sie die Ausgabe.

Vorgehensweise

Editieren userid.KURS.JCL($JOBCARD) mit Inhalt (ähnlich) wie unten angegeben

```
//RZSRGEN JOB (000947,TEST,,,,9900),
//         'ZIT P 2.21,SEIDLER',MSGCLASS=T,CLASS=G,
//         NOTIFY=RZSR,REGION=1M,TIME=(,30)
```

2.4 Jobstep erstellen

Member: JCL201

Erweitern Sie die Jobkarte um den Parameter MSGLEVEL.
Fügen Sie einen Step ein mit dem Programm IEFBR14.

Lassen Sie den Job mit unterschiedlichen Angaben für MSGLEVEL laufen und analysieren Sie die Ausgabe.

Vorgehensweise

Editieren userid.KURS.JCL(JCL201); kopieren Member $JOBCARD; Zeilen 4 und 5 mit Inhalt (ähnlich) wie unten angegeben; je nach Subparameter für MSGLEVEL wird die JCL unterschiedlich ausführlich protokolliert. Sinnvolle Angabe: MSGLEVEL=(1,1)

```
//RZSRGEN JOB (000947,TEST,,,,9900),
//        'ZIT P 2.21,SEIDLER',MSGCLASS=T,CLASS=G,
//        NOTIFY=&SYSUID,MSGLEVEL=(1,1)
//***  IEFBR14 MIT VERSCHIEDENEN MSGLEVEL
//STEP01  EXEC PGM=IEFBR14
```

2.5 Job mit 2 Steps

Member: JCL202

Erweitern Sie den Job JCL201 um einen zweiten Step, der ebenfalls das Programm IEFBR14 aufruft.

Lassen Sie den Job laufen und prüfen Sie die Ausgabe.
Was hat sich gegenüber der Ausgabe von JCL201 verändert?
Woran erkennen Sie, dass der 2. Step gelaufen ist?

Vorgehensweise

Editieren userid.KURS.JCL(JCL202); kopieren Member JCL201; weitere Zeile mit Inhalt (ähnlich) wie unten angegeben (STEP02); je nach Firma wird im Jobprotokoll für jeden Step eine Zeile protokolliert, in dem der RC beschrieben ist.

```
//RZSRGEN JOB (000947,TEST,,,,9900),
//        'ZIT P 2.21,SEIDLER',MSGCLASS=T,CLASS=G,
//        NOTIFY=&SYSUID,MSGLEVEL=(1,1)
//***  IEFBR14 MIT VERSCHIEDENEN MSGLEVEL
//STEP01  EXEC PGM=IEFBR14
//STEP02  EXEC PGM=IEFBR14
```

Beispielausgabe:

```
14.00.04 JOB09844  -JOBNAME  STEPNAME PROCSTEP    RC   EXCP    CPU    SRB  CLOCK
14.00.04 JOB09844  -RZSRGEN  STEP01               00      0    .00    .00     .0
14.00.04 JOB09844  -RZSRGEN  STEP02               00      0    .00    .00     .0
```

2.6 Job mit COND-Steuerung

Member: JCL203:

Erweitern Sie den Job JCL202 so, dass der 2. Step bei der Ausführung des Jobs nicht angestoßen wird.

Lassen Sie den Job laufen und prüfen Sie die Ausgabe.
Was hat sich gegenüber der Ausgabe von JCL202 verändert?
Woran erkennen Sie, dass der 2. Step nicht gelaufen ist?
Testen Sie den Job mit unterschiedlichen Angaben für die COND-Steuerung.

Vorgehensweise

Editieren userid.KURS.JCL(JCL203); kopieren Member JCL202; ändern letzte Zeile mit Inhalt (ähnlich) wie unten angegeben; je nach Firma wird im Jobprotokoll für jeden Step eine Zeile protokolliert, in dem der RC beschrieben ist.

```
//RZSRGEN JOB (000947,TEST,,,,9900),
//        'ZIT P 2.21,SEIDLER',MSGCLASS=T,CLASS=G,
//        NOTIFY=&SYSUID,MSGLEVEL=(1,1)
//***  IEFBR14 MIT VERSCHIEDENEN MSGLEVEL
//STEP01  EXEC PGM=IEFBR14
//STEP02  EXEC PGM=IEFBR14,COND=(0,EQ)
```

Beispielausgabe:

```
14.02.29 JOB09849  -JOBNAME  STEPNAME PROCSTEP    RC   EXCP    CPU    SRB  CLOCK
14.02.29 JOB09849  -RZSRGEN  STEP01              00      0    .00    .00     .0
14.02.29 JOB09849  -RZSRGEN  STEP02            FLUSH     0    .00    .00     .0
14.02.29 JOB09849  IEF404I RZSRGEN - ENDED - TIME=14.02.29
```

3 Arbeiten mit Dateien – 1

3.1 Lesen einer Instream-Datei

Member: JCL301

Erstellen Sie einen Job, der eine Instream-Datei einliest und auf die Spool ausgibt.
Nutzen Sie das Programm IEBGENER. Die DD-Anweisung für die Eingabe lautet SYSUT1, für die Ausgabe SYSUT2.
SYSIN DD DUMMY

Lassen Sie den Job laufen und prüfen Sie die Ausgabe.

Vorgehensweise

Editieren userid.KURS.JCL(JCL301); kopieren Member $JOBCARD; Zeilen mit Inhalt (ähnlich) wie unten angegeben.

```
//RZSRGEN JOB (000947,TEST,,,,9900),
//        'ZIT P 2.21,SEIDLER',MSGCLASS=T,CLASS=G,
//        NOTIFY=RZSR,MSGLEVEL=(1,1)
//*** LESEN INSTREAM; AUSGABE AUF SPOOL
//STEP01   EXEC PGM=IEBGENER
//SYSUT1   DD   *
DIES IST EINE ZEILE.
//SYSUT2   DD   SYSOUT=*
//SYSPRINT DD   SYSOUT=*
//SYSIN    DD   DUMMY
```

Beispielausgabe:

```
...
07.49.25 JOB11019  -JOBNAME  STEPNAME PROCSTEP    RC   EXCP    CPU    SRB  CLOCK
07.49.25 JOB11019  -RZSRGEN  STEP01               00     13    .00    .00     .0
07.49.25 JOB11019  IEF404I RZSRGEN - ENDED - TIME=07.49.25
...
IEF375I  JOB/RZSRGEN /START 2005212.0749
IEF376I  JOB/RZSRGEN /STOP  2005212.0749 CPU    0MIN 00.01SEC SRB    0MIN 00.00S
DIES IST EINE ZEILE.
DATA SET UTILITY - GENERATE
IEB352I WARNING: ONE OR MORE OF THE OUTPUT DCB PARMS COPIED FROM INPUT

PROCESSING ENDED AT EOD
******************************** BOTTOM OF DATA ********************************
```

3.2 Lesen einer Dummy-Datei

Member: JCL302

Erstellen Sie einen Job, der eine leere Datei, eine Dummy-Datei, liest und ausgibt.
Nutzen Sie das Programm IEBGENER.

Lassen Sie den Job laufen und prüfen Sie die Ausgabe.
Woran erkennen Sie, dass das Programm richtig gelaufen ist?

Vorgehensweise

Editieren userid.KURS.JCL(JCL302); kopieren Member JCL301; Zeilen mit Inhalt (ähnlich) wie unten ändern.

```
//RZSRGEN JOB (000947,TEST,,,,9900),
//         'ZIT P 2.21,SEIDLER',MSGCLASS=T,CLASS=G,
//         NOTIFY=RZSR,MSGLEVEL=(1,1)
//*** LESEN INSTREAM-DUMMY; AUSGABE AUF SPOOL
//STEP01   EXEC PGM=IEBGENER
//SYSUT1   DD   DUMMY
//SYSUT2   DD   SYSOUT=*
//SYSPRINT DD   SYSOUT=*
//SYSIN    DD   DUMMY
```

Beispielausgabe:

```
...
07.53.47 JOB11025  -JOBNAME  STEPNAME PROCSTEP    RC   EXCP    CPU    SRB  CLOCK
07.53.47 JOB11025  -RZSRGEN  STEP01               00      9    .00    .00     .0
07.53.47 JOB11025  IEF404I RZSRGEN - ENDED - TIME=07.53.47
...
DATA SET UTILITY - GENERATE
IEB352I WARNING: ONE OR MORE OF THE OUTPUT DCB PARMS COPIED FROM INPUT

PROCESSING ENDED AT EOD
******************************** BOTTOM OF DATA ********************************
```

Der Return-Code (0) gibt den Hinweis, dass der Step korrekt gelaufen ist.

3.3 Schreiben in das „Nirwana“

Member: JCL303

Erstellen Sie einen Job, der eine Instream-Datei einliest und die Ausgabe auf Dummy schreibt.
Nutzen Sie das Programm IEBGENER.

Lassen Sie den Job laufen und prüfen Sie die Ausgabe.
Woran erkennen Sie, dass das Programm richtig gelaufen ist?

Vorgehensweise

Editieren userid.KURS.JCL(JCL303); kopieren Member JCL301; Zeile mit SYSUT2 wie unten ändern.

```
//RZSRGEN JOB (000947,TEST,,,,9900),
//        'ZIT P 2.21,SEIDLER',MSGCLASS=T,CLASS=G,
//        NOTIFY=RZSR,MSGLEVEL=(1,1)
//*** LESEN INSTREAM-DUMMY; AUSGABE AUF SPOOL
//STEP01   EXEC PGM=IEBGENER
//SYSUT1   DD   DUMMY
//SYSUT2   DD   DUMMY
//SYSPRINT DD   SYSOUT=*
//SYSIN    DD   DUMMY
```

Beispielausgabe:

```
...
08.00.29 JOB11042  -JOBNAME  STEPNAME PROCSTEP    RC   EXCP    CPU    SRB  CLOCK
08.00.29 JOB11042  -RZSRGEN  STEP01               00      8    .00    .00     .0
08.00.29 JOB11042  IEF404I RZSRGEN - ENDED - TIME=08.00.29
...
DATA SET UTILITY - GENERATE
IEB352I WARNING: ONE OR MORE OF THE OUTPUT DCB PARMS COPIED FROM INPUT

PROCESSING ENDED AT EOD
****************************** BOTTOM OF DATA ******************************
```

Der Return-Code (0) gibt den Hinweis, dass der Step korrekt gelaufen ist.

4 Arbeiten mit Dateien – 2

4.1 Anlegen einer sequentiellen Datei

Member: JCL401

Erstellen Sie einen Job, der eine sequentielle Datei userid.TEST.PS anlegt.
Die Datei hat eine feste Länge von 80 Stellen und ist geblockt.
Wählen Sie die Angaben so, dass mindestens 1000 Zeilen in diese Datei passen.

Lassen Sie den Job laufen und prüfen Sie die Ausgabe.
Woran erkennen Sie, dass die Datei richtig angelegt worden ist?

Gehen Sie anschließend im EDIT in die Datei und schreiben Sie einen beliebigen Inhalt hinein. Testen Sie, das mindestens 1000 Zeilen, aber nicht mehr als 1500 Zeilen in der Datei Platz haben.
Wie prüfen Sie die Grenze des Erreichbaren?

Vorgehensweise

Editieren userid.KURS.JCL(JCL401); kopieren Member $JOBCARD; Zeilen mit Inhalt wie unten ergänzen. Dabei wurde schon vorgesehen, dass eine eventuelle schon existierende Datei gleichen Namens vorab gelöscht wird.

```
//RZSRGEN JOB (000947,TEST,,,,9900),
//         'ZIT P 2.21,SEIDLER',MSGCLASS=T,CLASS=G,
//        NOTIFY=RZSR,MSGLEVEL=(1,1)
//*** ERSTELLEN DATEI MIT 80 STELLEN FUER 1000 ZEILEN
//*** EVENTUELL EXISTIERENDE DATEI VORHER LOESCHEN
//RESTART  EXEC PGM=IEFBR14
//ALT      DD DSN=&SYSUID..TEST.PS,DISP=(MOD,DELETE,DELETE),
//            DCB=(LRECL=80,RECFM=FB,BLKSIZE=0),
//            SPACE=(TRK,(3,0)),UNIT=SYSDA
//*** ERSTELLEN DATEI MIT 80 STELLEN FUER 1000 ZEILEN
//STEP01   EXEC PGM=IEFBR14
//NEU      DD DSN=&SYSUID..TEST.PS,DISP=(NEW,CATLG,DELETE),
//            DCB=(LRECL=80,RECFM=FB,BLKSIZE=0),
//            SPACE=(TRK,(3,0)),UNIT=SYSDA
```

Beispielausgabe:

```
...
08.15.55 JOB11071  -JOBNAME  STEPNAME PROCSTEP    RC   EXCP    CPU    SRB  CLOCK
08.15.55 JOB11071  -RZSRGEN  RESTART              00      3    .00    .00     .0
08.15.55 JOB11071  -RZSRGEN  STEP01               00      0    .00    .00     .0
...
       3 //ALT      DD DSN=&SYSUID..TEST.PS,DISP=(MOD,DELETE,DELETE),
         //            DCB=(LRECL=80,RECFM=FB,BLKSIZE=0),
         //            SPACE=(TRK,(3,0)),UNIT=SYSDA
         //*** ERSTELLEN DATEI MIT 80 STELLEN FUER 1000 ZEILEN
         IEFC653I SUBSTITUTION JCL - DSN=RZSR.TEST.PS,DISP=(MOD,DELETE,DELETE),
         SPACE=(TRK,(3,0)),UNIT=SYSDA
...
IEF236I ALLOC. FOR RZSRGEN RESTART
IGD101I SMS ALLOCATED TO DDNAME (ALT     )
        DSN (RZSR.TEST.PS                                  )
        STORCLAS (USRSTD) MGMTCLAS (STANDARD) DATACLAS (DMYDSORG)
        VOL SER NOS= USR023
IEF142I RZSRGEN RESTART - STEP WAS EXECUTED - COND CODE 0000
IGD105I RZSR.TEST.PS                                DELETED,   DDNAME=ALT
...
IEF236I ALLOC. FOR RZSRGEN STEP01
IGD101I SMS ALLOCATED TO DDNAME (NEU     )
        DSN (RZSR.TEST.PS                                  )
        STORCLAS (USRSTD) MGMTCLAS (STANDARD) DATACLAS (DMYDSORG)
        VOL SER NOS= USR020
IEF142I RZSRGEN STEP01 - STEP WAS EXECUTED - COND CODE 0000
IGD104I RZSR.TEST.PS                                RETAINED,  DDNAME=NEU
...
```

Hinweise:

- „`IEFC653I SUBSTITUTION JCL` “ zeigt an, was aus der Variabel &SYSUID gemacht worden ist.
- „`DSN (RZSR.TEST.PS`“ … die Datei wird temporär angelegt
- „`RZSR.TEST.PS  DELETED`“ … die Datei wird gelöscht
- „`RZSR.TEST.PS  RETAINED`“ … die Datei ist angelegt

Beim Editieren der Datei taucht der folgende Fehler auf (oder ein ähnlicher wie SB37):

```
IEA705I ERROR DURING GETMAIN SYS CODE = 878-10 RZSR T8 LOGON2T8 00
IEA705I 00F44700 007DC548 007DC548 00007200 0001FFE0
***

oder:
IEC031I D37-04,IFG0554P,RZSR,T8,ISP07390,1213,USR020,RZSR.TEST.PS
***
```

Der Fehler taucht erst beim Speichern der Datei auf. Eventuelle Compressionverfahren können den benötigten Speicher verringern. Eine *exakte* Berechnung (1 Spur mit 47kB) gibt es also nicht.

4.2 Anlegen einer PO-Datei

Member: JCL402

Erstellen Sie einen Job, der eine PO-Datei userid.TEST.PDS anlegt. Die Datei hat eine feste Länge von 80 Stellen und ist geblockt.
Wählen Sie die Angaben so, dass mindestens 1000 Zeilen in die einzelnen Member passen und mindestens 20 Member in das Directory aufgenommen werden können.

Lassen Sie den Job laufen und prüfen Sie die Ausgabe.
Überprüfen Sie die Dateidefinition an Hand der Ausgabe des Jobs und auf einem anderen beliebigen Weg Ihrer Wahl.

Vorgehensweise

Editieren userid.KURS.JCL(JCL402); kopieren Member JCL401; Zeilen mit Inhalten wie unten ersetzen. Dabei wurde schon vorgesehen, dass eine eventuelle schon existierende Datei gleichen Namens vorab gelöscht wird.

```
//RZSRGEN JOB (000947,TEST,,,,9900),
//         'ZIT P 2.21,SEIDLER',MSGCLASS=T,CLASS=G,
//        NOTIFY=RZSR,MSGLEVEL=(1,1)
//*** ERSTELLEN DATEI MIT 80 STELLEN FUER 1000 ZEILEN
//*** EVENTUELL EXISTIERENDE DATEI VORHER LOESCHEN
//RESTART  EXEC PGM=IEFBR14
//ALT      DD DSN=&SYSUID..TEST.PDS,DISP=(MOD,DELETE,DELETE),
//            DCB=(LRECL=80,RECFM=FB,BLKSIZE=0),
//            SPACE=(TRK,(3,0,5)),UNIT=SYSDA
//*** ERSTELLEN DATEI MIT 80 STELLEN FUER 1000 ZEILEN
//STEP01   EXEC PGM=IEFBR14
//NEU      DD DSN=&SYSUID..TEST.PDS,DISP=(NEW,CATLG,DELETE),
//            DCB=(LRECL=80,RECFM=FB,BLKSIZE=0),
//            SPACE=(TRK,(3,0,5)),UNIT=SYSDA
```

Beispielausgabe:

Die Ausgabe sieht aus wie bei JCL401.

Größe der Directory: für ca. 4-5 Member wird ein Directory-Track benötigt.

```
EDIT       RZSR.TEST.PDS(A30) - 01.00                  No space in directory
Command ===>                                               Scroll ===> CSR
```

Bei meinem Test kam bei Speichern des 30. Members eine Fehlermeldung.

Achtung: vor der nächsten Übung müssen die „überflüssigen“ Member wieder gelöscht und ein compress gemacht werden (ISPF 3.1).

4.3 Kopieren in ein Member einer PO-Datei

Member. JCL403

Erstellen Sie einen Job, der den von Ihnen erzeugten Inhalt der Datei aus Aufgabe 4.1 in das Member AUSSEQ der PO-Datei userid.TEST.PDS überträgt.

Lassen Sie den Job laufen und überprüfen Sie die Ausgabe und das Ergebnis.

Vorgehensweise

Editieren userid.KURS.JCL(JCL403); kopieren Member JCL301; Zeilen mit Inhalten wie unten ersetzen. Eventuell muss auf die Eingabedatei verkleinert werden (ISPF-Edit) und der unbenutzte Platz freigegeben werden (ISPF 3.4 Funktion F), ehe der Job fehlerfrei laufen kann.

```
//RZSRGEN JOB (000947,TEST,,,,9900),
//        'ZIT P 2.21,SEIDLER',MSGCLASS=T,CLASS=G,
//        NOTIFY=RZSR,MSGLEVEL=(1,1)
//*** LESEN INSTREAM; AUSGABE AUF SPOOL
//STEP01   EXEC PGM=IEBGENER
//SYSUT1   DD DISP=SHR,DSN=&SYSUID..TEST.PS
//SYSUT2   DD DISP=SHR,DSN=&SYSUID..TEST.PDS(AUSSEQ)
//SYSPRINT DD   SYSOUT=*
//SYSIN    DD   DUMMY
```

Zu beachten ist, dass auf der Ausgabedatei mit DISP=SHR gearbeitet werden muss. DISP=NEW heißt, dass die *Datei* neu ist, nicht ein Member. Die Neuerstellung von Membern übernimmt die PO-Datei selbst.

4.4 Kopieren PO-Member

Member: JCL404

Erstellen Sie einen Job, der das Member AUSSEQ der PO-Datei userid.TEST.PDS in das Member AUSSEQ2 der gleichen PO-Datei kopiert.
Geht das überhaupt?
Kann das auch verhindert werden?

Lassen Sie den Job laufen und prüfen Sie die Ausgabe und das Ergebnis.

Vorgehensweise

Editieren userid.KURS.JCL(JCL404); kopieren Member JCL403; Zeilen mit SYSUT1 und SYSUT2 wie unten ändern.

```
//RZSRGEN JOB (000947,TEST,,,,9900),
//         'ZIT P 2.21,SEIDLER',MSGCLASS=T,CLASS=G,
//         NOTIFY=RZSR,MSGLEVEL=(1,1)
//*** LESEN INSTREAM; AUSGABE AUF SPOOL
//STEP01   EXEC PGM=IEBGENER
//SYSUT1   DD DISP=SHR,DSN=&SYSUID..TEST.PDS(AUSSEQ)
//SYSUT2   DD DISP=SHR,DSN=&SYSUID..TEST.PDS(AUSSEQ1)
//SYSPRINT DD   SYSOUT=*
//SYSIN    DD   DUMMY
```

Anzeige des Directory der Zieldatei:

```
EDIT     RZSR.TEST.PDS                                    Row 00001 of 00003
Command ===>                                                Scroll ===> CSR
   Name     Prompt          Size    Created           Changed             ID
. AUSSEQ
. AUSSEQ1
. A01                          2   2005/07/31   2005/07/31 08:34:11    RZSR
  **End**
```

5 Utilities

Keine Übungen bzw. siehe Unterlagen zu Kurs z/OS Dienstprogramme

6 Jobsteuerung, Stepsteuerung

6.1 Anlegen einer sequentiellen Datei mit Rückbezug

Member: JCL601

Erstellen Sie einen Job, dessen erster Step die sequentielle Datei userid.TEST.PS11 erzeugt. Die Datei hat eine feste Satzlänge von 80 Bytes und ist geblockt. Die Datei soll mindestens 1000 Zeilen aufnehmen können.
Erstellen Sie einen weiteren Step, der die sequentielle Datei userid.TEST.PS12 erzeugt. Die Angaben für die Datei sollen über einen Rückbezug beschrieben werden.

Lassen Sie den Job laufen und prüfen Sie die Ausgabe und das Ergebnis.

Schreiben Sie im Edit einige Zeilen in die Datei userid.TEST.PS11.

Vorgehensweise

Editieren userid.KURS.JCL(JCL601); kopieren Member JCL401; Zeilen wie unten angegeben (oder ähnlich) ändern.

```
//RZSRGEN JOB (000947,TEST,,,,9900),
//         'ZIT P 2.21,SEIDLER',MSGCLASS=T,CLASS=G,
//        NOTIFY=RZSR,MSGLEVEL=(1,1)
//*** ERSTELLEN DATEI MIT 80 STELLEN FUER 1000 ZEILEN
//*** ERSTELLEN DATEI MIT RUECKBEZUG
//*** EVENTUELL EXISTIERENDE DATEI VORHER LOESCHEN
//RESTART  EXEC PGM=IEFBR14
//ALT01    DD DSN=&SYSUID..TEST.PS11,DISP=(MOD,DELETE,DELETE),
//            DCB=(LRECL=80,RECFM=FB,BLKSIZE=0),
//            SPACE=(TRK,(3,0)),UNIT=SYSDA
//ALT02    DD DSN=&SYSUID..TEST.PS12,DISP=(MOD,DELETE,DELETE),
//            DCB=(LRECL=80,RECFM=FB,BLKSIZE=0),
//            SPACE=(TRK,(3,0)),UNIT=SYSDA
//*** ERSTELLEN DATEI MIT 80 STELLEN FUER 1000 ZEILEN
//STEP01   EXEC PGM=IEFBR14
//NEU      DD DSN=&SYSUID..TEST.PS11,DISP=(NEW,CATLG,DELETE),
//            DCB=(LRECL=80,RECFM=FB,BLKSIZE=0),
//            SPACE=(TRK,(3,0)),UNIT=SYSDA
//*** ERSTELLEN DATEI MIT RUECKBEZUG
//STEP02   EXEC PGM=IEFBR14
//NEU      DD DSN=&SYSUID..TEST.PS12,DISP=(NEW,CATLG,DELETE),
//            DCB=*.STEP01.NEU,
//            SPACE=(TRK,(3,0)),UNIT=SYSDA
```

6.2 Kopieren einer Datei mit eventueller Neuanlage

Member: JCL602

Erstellen Sie einen Job mit dem folgenden Ablauf:

- Anlegen sequentielle Datei userid.TEST.PS21 (Angaben wie oben)
- Falls Anlegen nok
  - Löschen Datei userid.TEST.PS21
  - Datei userid.TEST.PS21 neu anlegen (Angaben wie oben)
- Kopieren Inhalt von userid.TEST.PS11 nach userid.TEST.PS21

Lassen Sie den Job laufen und prüfen Sie die Ausgabe und das Ergebnis.

Vorgehensweise

Editieren userid.KURS.JCL(JCL602); kopieren Member JCL401; Zeilen wie unten angegeben (oder ähnlich) ändern.

```
//RZSRGEN JOB (000947,TEST,,,,9900),
//        'ZIT P 2.21,SEIDLER',MSGCLASS=T,CLASS=G,
//        NOTIFY=RZSR,MSGLEVEL=(1,1)
//*** ERSTELLEN DATEI MIT 80 STELLEN FUER 1000 ZEILEN
//STEP01   EXEC PGM=IEFBR14
//NEU      DD DSN=&SYSUID..TEST.PS21,DISP=(NEW,CATLG,DELETE),
//            DCB=(LRECL=80,RECFM=FB,BLKSIZE=0),
//            SPACE=(TRK,(3,3)),UNIT=SYSDA
//*** FALLS ERSTELLEN NICHT ERFOLGREICH, ERST LOESCHEN
//IF01     IF (STEP01.ABEND) THEN
//*** DATEI LOESCHEN
//RESTART  EXEC PGM=IEFBR14
//ALT01    DD DSN=&SYSUID..TEST.PS11,DISP=(MOD,DELETE,DELETE),
//            DCB=(LRECL=80,RECFM=FB,BLKSIZE=0),
//            SPACE=(TRK,(3,3)),UNIT=SYSDA
//*** ERSTELLEN DATEI MIT RUECKBEZUG
//STEP02   EXEC PGM=IEFBR14
//NEU      DD DSN=&SYSUID..TEST.PS12,DISP=(NEW,CATLG,DELETE),
//            DCB=(LRECL=80,RECFM=FB,BLKSIZE=0),
```

Dieser Job läuft beim ersten Mal wie gewünscht. Doch der gewollte Effekt, dass beim zweiten Durchlauf der If-Zweig genutzt wird, klappt nicht. Der Grund liegt in den 3 Phasen beim Jobdruchlauf:
1. Phase: Syntaxcheck
2. Phase: Umgebung prüfen und bereit stellen
3. Phase: Job läuft – und erst hier kann er die IFs interpretieren und umsetzen
In der 2. Phase erkennt aber JES, dass die Datei schon vorhanden ist und bricht ab.
Lösung: siehe vorherige Aufgaben, in der mit (MOD,DELETE,DELETE) im 1. Step gearbeitet wird.

7 Jobs mit Ladebibliotheken

7.1 PO-Datei als Ladebibliothek anlegen

Member: JCL701

Erstellen Sie einen Job der eine Ladebibliothek mit dem Namen userid.TEST.LOAD erstellt. Holen Sie sich die Parameter aus einer Systembibliothek.

Lassen Sie den Job laufen und prüfen Sie die Ausgabe und das Ergebnis.
Kann diese Datei auch über ISPF 3.2 erzeugt werden?
Kopieren Sie ein beliebiges Programm aus einer Systembibliothek hinein.

Vorgehensweise

Editieren userid.KURS.JCL(JCL701); kopieren Member JCL401; Zeilen wie unten angegeben (oder ähnlich) ändern. Die DCB-Angaben holt man sich aus der Definition einer Ladebibliothek wie SYS1.LINKLIB o.Ä..

```
//RZSRGEN JOB (000947,TEST,,,,9900),
//         'ZIT P 2.21,SEIDLER',MSGCLASS=T,CLASS=G,
//        NOTIFY=RZSR,MSGLEVEL=(1,1)
//*** ERSTELLEN LADEBIBLIOTHEK
//STEP01   EXEC PGM=IEFBR14
//NEU      DD DSN=&SYSUID..TEST.LOAD,DISP=(NEW,CATLG,DELETE),
//            DCB=(LRECL=0,RECFM=U,BLKSIZE=32760),
//            SPACE=(TRK,(20,0,20)),UNIT=SYSDA
```

Nach korrektem Joblauf aus der Systembibliothek das Member IEBGENER über 3.3 kopieren.

7.2 Job mit STEPLIB

Member: JCL702

Erstellen Sie einen Job, der auf ein / das Programm aus der Bibliothek userid.TEST.LOAD zugreift. Benutzen Sie eine STEPLIB-Anweisung.

Lassen Sie den Job laufen und prüfen Sie die Ausgabe und das Ergebnis. Woran können Sie erkennen, dass Sie Ihre Bibliothek und nicht die Systembibliothek benutzt haben?

Vorgehensweise

Editieren userid.KURS.JCL(JCL702); kopieren Member JCL301; Zeile mit STEPLIB einfügen.

```
//RZSRGEN JOB (000947,TEST,,,,9900),
//         'ZIT P 2.21,SEIDLER',MSGCLASS=T,CLASS=G,
//         NOTIFY=RZSR,MSGLEVEL=(1,1)
//*** LESEN INSTREAM; AUSGABE AUF SPOOL - MIT STEPLIB
//STEP01   EXEC PGM=IEBGENER
//STEPLIB  DD DISP=SHR,DSN=&SYSUID..TEST.LOAD
//SYSUT1   DD *
DIES IST EINE ZEILE.
//SYSUT2   DD SYSOUT=*
//SYSPRINT DD SYSOUT=*
//SYSIN    DD DUMMY
```

Es gibt im Job keinerlei Hinweise darauf, aus welcher Ladebibliothek das Programm geladen worden ist. Der Hinweis …

```
IGD104I RZSR.TEST.LOAD                                RETAINED,  DDNAME=STEPLIB
```

… heißt nur, dass die Datei gefunden wurde, sonst nichts. Testen kann man dies, wenn man ein Programm aufruft, das nicht in der Bibliothek vorhanden ist; die Jobausgabe ändert sich dadurch nicht. Ein weiterer Test ist möglich, wenn das Programm mit einem Fantasienamen versehen wird, der mit Sicherheit nicht im Systemvorhanden ist.

7.3 Job mit JOBLIB

Member: JCL703

Erstellen Sie einen Job, der auf ein / das Programm aus der Bibliothek userid.TEST.LOAD zugreift. Benutzen Sie eine JOBLIB-Anweisung.

Lassen Sie den Job laufen und prüfen Sie die Ausgabe und das Ergebnis. Woran können Sie erkennen, dass Sie Ihre Bibliothek und nicht die Systembibliothek benutzt haben?

Vorgehensweise

Editieren userid.KURS.JCL(JCL703); kopieren Member JCL702; Zeile mit STEPLIB entfernen und Zeile mit JOBLIB einfügen.

```
//RZSRGEN JOB (000947,TEST,,,,9900),
//         'ZIT P 2.21,SEIDLER',MSGCLASS=T,CLASS=G,
//         NOTIFY=RZSR,MSGLEVEL=(1,1)
//*** LESEN INSTREAM; AUSGABE AUF SPOOL - MIT STEPLIB
//JOBLIB   DD DISP=SHR,DSN=&SYSUID..TEST.LOAD
//STEP01   EXEC PGM=IEBGENER
//SYSUT1   DD *
DIES IST EINE ZEILE.
//SYSUT2   DD SYSOUT=*
//SYSPRINT DD SYSOUT=*
//SYSIN    DD DUMMY
```

Es gibt auch in diesem Fall im Job keinerlei Hinweise darauf, aus welcher Ladebibliothek das Programm geladen worden ist.

7.4 Job mit falschem Programmnamen

Member: JCL704

Erstellen Sie einen Job, der ein nicht vorhandenes Programm aufruft.

Lassen Sie den Job laufen und prüfen Sie die Ausgabe.
Woran können Sie in der Jobausgabe den falschen Namen erkennen?

Vorgehensweise

Editieren userid.KURS.JCL(JCL704); kopieren Member JCL703; Zeile mit Programmname ändern. Evtl. Zeile mit Joblib entfernen.

```
//RZSRGEN JOB (000947,TEST,,,,9900),
//        'ZIT P 2.21,SEIDLER',MSGCLASS=T,CLASS=G,
//       NOTIFY=RZSR,MSGLEVEL=(1,1)
//*** LESEN INSTREAM; AUSGABE AUF SPOOL - MIT FALSCHEM PGMNAME
//JOBLIB   DD DISP=SHR,DSN=&SYSUID..TEST.LOAD
//STEP01   EXEC PGM=OTTOHUGO
//SYSUT1   DD *
DIES IST EINE ZEILE.
//SYSUT2   DD SYSOUT=*
//SYSPRINT DD SYSOUT=*
//SYSIN    DD DUMMY
```

Beispielausgabe auf Bildschirm (durch NOTIFY ausgelöst):

```
09.54.19 JOB11268 $HASP165 RZSRGEN  ENDED AT NTZD - ABENDED S806 U0000 CN(INTERNAL)
***
```

Auszug aus Job:

```
...
09.54.18 JOB11268  CSV003I REQUESTED MODULE OTTOHUGO NOT FOUND
09.54.18 JOB11268  CSV028I ABEND806-04  JOBNAME=RZSRGEN   STEPNAME=STEP01
09.54.18 JOB11268  IEA995I SYMPTOM DUMP OUTPUT  656
   656             SYSTEM COMPLETION CODE=806  REASON CODE=00000004
   656              TIME=09.54.18  SEQ=02587  CPU=0000  ASID=0219
   656              PSW AT TIME OF ERROR  070C1000   811EB88E  ILC 2  INTC 0D
   656                NO ACTIVE MODULE FOUND
   656                NAME=UNKNOWN
   656                DATA AT PSW  011EB888 - 9024181E  0A0D18FB  180C181D
...
   656                      E: 00000000/84806000   F: 00000000/00000004
   656              END OF SYMPTOM DUMP
09.54.18 JOB11268  IEF450I RZSRGEN STEP01 - ABEND=S806 U0000 REASON=00000004  66
   660                     TIME=09.54.18
09.54.18 JOB11268  -                                          --TIMINGS (MINS.)--
09.54.19 JOB11268  -JOBNAME  STEPNAME PROCSTEP    RC   EXCP    CPU    SRB  CLOCK
09.54.19 JOB11268  -RZSRGEN  STEP01           *S806      5    .00    .00     .0
09.54.19 JOB11268  IEF404I RZSRGEN - ENDED - TIME=09.54.19
...
```

7.5 Anlegen GDG-Base-Entry

Member: JCL705

Erstellen Sie einen Job, der ein GDG-Base-Entry mit 3 Generationen erzeugt.

Lassen Sie den Job laufen und prüfen Sie die Ausgabe und das Ergebnis.

Vorgehensweise

Editieren userid.KURS.JCL(JCL705); kopieren Member $JOBCARD; Zeilen für IDCAMS wie unten ergänzen.

```
//RZSRGEN JOB (000947,TEST,,,,9900),
//        'ZIT P 2.21,SEIDLER',MSGCLASS=T,CLASS=G,
//        NOTIFY=RZSR,MSGLEVEL=(1,1)
//*** ANLEGEN GDG BASE ENTRY
//STEP01   EXEC PGM=IDCAMS
//SYSPRINT DD SYSOUT=*
//SYSIN    DD *
  DEFINE GDG (NAME(RZSR.TEST.GDG)  LIMIT(3) SCRATCH)
```

Beispielausgabe:

```
...
10.06.15 JOB11331  -JOBNAME  STEPNAME PROCSTEP    RC   EXCP    CPU    SRB  CLOCK
10.06.15 JOB11331  -RZSRGEN  STEP01               00      8    .00    .00     .0
10.06.15 JOB11331  IEF404I RZSRGEN - ENDED - TIME=10.06.15
...
IDCAMS  SYSTEM SERVICES                                        TIME: 10:06:15

  DEFINE GDG (NAME(RZSR.TEST.GDG)  LIMIT(3) SCRATCH)
IDC0001I FUNCTION COMPLETED, HIGHEST CONDITION CODE WAS 0

IDC0002I IDCAMS PROCESSING COMPLETE. MAXIMUM CONDITION CODE WAS 0
```

7.6 Kopieren Daten auf GDS

Member: JCL706

Erstellen Sie einen Job, der Daten auf eine neue Generation des GDG kopiert. Benutzen Sie als Eingabe die Datei userid.TEST.PS11.

Lassen Sie den Job laufen und prüfen Sie die Ausgabe und das Ergebnis. Lassen Sie den Job mindestens 3 Mal laufen und prüfen Sie jeweils die Ergebnisse.

Vorgehensweise

Editieren userid.KURS.JCL(JCL706); kopieren Member JCL403; Zeilen für SYSUT1 und SYSUT2 wie unten abändern.

```
//RZSRGEN JOB (000947,TEST,,,,9900),
//         'ZIT P 2.21,SEIDLER',MSGCLASS=T,CLASS=G,
//        NOTIFY=RZSR,MSGLEVEL=(1,1)
//*** LESEN INSTREAM; AUSGABE AUF SPOOL
//STEP01   EXEC PGM=IEBGENER
//SYSUT1   DD DISP=SHR,DSN=&SYSUID..TEST.PS11
//SYSUT2   DD DISP=(NEW,CATLG,DELETE),DSN=&SYSUID..TEST.GDG(+1),
//            DCB=(LRECL=80,RECFM=FB,BLKSIZE=0),
//            SPACE=(TRK,(3,3),RLSE),UNIT=SYSDA
//SYSPRINT DD   SYSOUT=*
//SYSIN    DD   DUMMY
```

Nach viermaligem Laufenlassen des Job erhält man über 3.4 die folgende Anzeige:

```
DSLIST - Data Sets Matching RZSR.TEST.GDG                       Row 1 of 4
Command ===>                                                 Scroll ===> CSR

Command - Enter "/" to select action              Message           Volume
-------------------------------------------------------------------------------
         RZSR.TEST.GDG                                                ??????
         RZSR.TEST.GDG.G0002V00                                       USR018
         RZSR.TEST.GDG.G0003V00                                       USR020
         RZSR.TEST.GDG.G0004V00                                       USR023
****************************** End of Data Set list ****************************
```

7.7 Arbeiten mit aktuellem GDS

Member: JCL707

Erstellen Sie einen Job, der die aktuelle Generation des GDG auf Spool ausgibt.

Lassen Sie den Job laufen und prüfen Sie die Ausgabe und das Ergebnis.

Vorgehensweise

Editieren userid.KURS.JCL(JCL707); kopieren Member JCL403; Zeilen für SYSUT1 und SYSUT2 wie unten abändern.

```
//RZSRGEN JOB (000947,TEST,,,,9900),
//         'ZIT P 2.21,SEIDLER',MSGCLASS=T,CLASS=G,
//        NOTIFY=RZSR,MSGLEVEL=(1,1)
//*** LESEN GDG AKTUELL; AUSGABE AUF SPOOL
//STEP01   EXEC PGM=IEBGENER
//SYSUT1   DD DISP=SHR,DSN=&SYSUID..TEST.GDG(0)
//SYSUT2   DD SYSOUT=*
//SYSPRINT DD SYSOUT=*
//SYSIN    DD DUMMY
```

7.8 Arbeiten mit allen Generationen einer GDG

Member JCL708

Erstellen Sie einen Job, der alle Generationen des GDG auf Spool ausgibt.

Lassen Sie den Job laufen und prüfen Sie die Ausgabe und das Ergebnis.

Vorgehensweise

Editieren userid.KURS.JCL(JCL708); kopieren Member JCL707; Zeile für SYSUT1 wie unten abändern.

```
//RZSRGEN JOB (000947,TEST,,,,9900),
//         'ZIT P 2.21,SEIDLER',MSGCLASS=T,CLASS=G,
//        NOTIFY=RZSR,MSGLEVEL=(1,1)
//*** LESEN GDG AKTUELL; AUSGABE AUF SPOOL
//STEP01   EXEC PGM=IEBGENER
//SYSUT1   DD DISP=SHR,DSN=&SYSUID..TEST.GDG
//SYSUT2   DD SYSOUT=*
//SYSPRINT DD SYSOUT=*
//SYSIN    DD DUMMY
```

In der Jobausgabe taucht die Zeile aus meiner Datei 3 Mal auf, da ich die Originaldatei 4 Mal jeweils in eine neue Generation kopiert habe und maximal 3 Generationen existieren.

```
IEF375I  JOB/RZSRGEN /START 2005212.1027
IEF376I  JOB/RZSRGEN /STOP  2005212.1027 CPU    0MIN 00.01SEC SRB    0MIN 00.00S
Dies ist eine Zeile von TEST.PS11.                                      00000100
Dies ist eine Zeile von TEST.PS11.                                      00000100
Dies ist eine Zeile von TEST.PS11.                                      00000100
DATA SET UTILITY - GENERATE
IEB352I WARNING: ONE OR MORE OF THE OUTPUT DCB PARMS COPIED FROM INPUT

PROCESSING ENDED AT EOD
```

7.9 Löschen GDG

Member: JCL709

Erstellen Sie einen Job, der die GDG mit allen GDS löscht.

Lassen Sie den Job laufen und prüfen Sie die Ausgabe und das Ergebnis.

Vorgehensweise

Editieren userid.KURS.JCL(JCL709); kopieren Member JCL705; Zeile für IDCAMS wie unten abändern.

```
//RZSRGEN JOB (000947,TEST,,,,9900),
//        'ZIT P 2.21,SEIDLER',MSGCLASS=T,CLASS=G,
//        NOTIFY=RZSR,MSGLEVEL=(1,1)
//*** ANLEGEN GDG BASE ENTRY
//STEP01   EXEC PGM=IDCAMS
//SYSPRINT DD SYSOUT=*
//SYSIN    DD *
  DELETE RZSR.TEST.GDG GDG RECOVERY
```

Achtung: Bei dieser Vorgehensweise wird das GDG-Base-Entry gelöscht und die GDS entkatalogisiert. Sie stehen daher noch physisch auf der Platte. Daher vor dem Löschen des Base-Entry immer die GDS löschen.

7.10 STORCLAS im Unternehmen

Prüfen Sie im ISPF, welche Storage-Klassen im Unternehmen existieren.

7.11 MGMTCLAS im Unternehmen

Prüfen Sie im ISPF, welche Management-Klassen im Unternehmen existieren.

8 Jobs mit Prozeduren

8.1 Prozeduraufruf - Instream

Member JCL801

Erstellen Sie einen Job auf der Basis von JCL401. Der Step von JCL401 soll als Instream-Prozedur aufgerufen werden.

Lassen Sie den Job laufen und prüfen Sie die Ausgabe und das Ergebnis.

Vorgehensweise

Editieren userid.KURS.JCL(JCL801); kopieren Member JCL401. Die zwei vorhandenen Steps / den vorhandenen Step in die „Klammer“ PROC / PEND einfassen. Aufrufstep für die Prozedur anhängen.

```
//RZSRGEN JOB (000947,TEST,,,,9900),
//         'ZIT P 2.21,SEIDLER',MSGCLASS=T,CLASS=G,
//        NOTIFY=RZSR,MSGLEVEL=(1,1)
//*** KOPIE VON JCL401 ALS PROZEDUR
//PROZ01   PROC
//*** EVENTUELL EXISTIERENDE DATEI VORHER LOESCHEN
//RESTART  EXEC PGM=IEFBR14
//ALT      DD DSN=&SYSUID..TEST.PS,DISP=(MOD,DELETE,DELETE),
//            DCB=(LRECL=80,RECFM=FB,BLKSIZE=0),
//            SPACE=(TRK,(3,0)),UNIT=SYSDA
//*** ERSTELLEN DATEI MIT 80 STELLEN FUER 1000 ZEILEN
//STEP01   EXEC PGM=IEFBR14
//NEU      DD DSN=&SYSUID..TEST.PS,DISP=(NEW,CATLG,DELETE),
//            DCB=(LRECL=80,RECFM=FB,BLKSIZE=0),
//            SPACE=(TRK,(3,0)),UNIT=SYSDA
//PROZ01E  PEND
//GO01     EXEC PROC=PROZ01
```

Im Job kommt ein Hinweis, wo die Prozedur gefunden worden ist:

```
STMT NO. MESSAGE
3 IEFC001I PROCEDURE PROZ01 WAS EXPANDED USING INSTREAM PROCEDURE DEFINITION
```

8.2 Prozeduraufruf – Instream, parametrisiert

Member JCL802

Erstellen Sie einen Job, der die vorige Prozedur mit einem Parameter aufruft. Parameter soll der Dateiname sein.

Lassen Sie den Job laufen und prüfen Sie die Ausgabe und das Ergebnis.

Vorgehensweise

Editieren userid.KURS.JCL(JCL802); kopieren Member JCL801. Feste Werte in DSN=… entfernen und mit Variablen bestücken. Bei Prozeduraufruf Wert für Variable DATEI angeben.

```
//RZSRGEN JOB (000947,TEST,,,,9900),
//         'ZIT P 2.21,SEIDLER',MSGCLASS=T,CLASS=G,
//        NOTIFY=RZSR,MSGLEVEL=(1,1)
//*** KOPIE VON JCL401 ALS PROZEDUR
//PROZ01   PROC DATEI=
//*** EVENTUELL EXISTIERENDE DATEI VORHER LOESCHEN
//RESTART  EXEC PGM=IEFBR14
//ALT      DD DSN=&DATEI,DISP=(MOD,DELETE,DELETE),
//            DCB=(LRECL=80,RECFM=FB,BLKSIZE=0),
//            SPACE=(TRK,(3,0)),UNIT=SYSDA
//*** ERSTELLEN DATEI MIT 80 STELLEN FUER 1000 ZEILEN
//STEP01   EXEC PGM=IEFBR14
//NEU      DD DSN=&DATEI,DISP=(NEW,CATLG,DELETE),
//            DCB=(LRECL=80,RECFM=FB,BLKSIZE=0),
//            SPACE=(TRK,(3,0)),UNIT=SYSDA
//PROZ01   PEND
//GO01     EXEC PROC=PROZ01,DATEI=&SYSUID..TEST.PS
```

Im Joboutput wird erwähnt, wo die Prozedur herkommt und er weist alle Zeilen aus, in denen Variablen ersetzt worden sind.

8.3 Prozeduraufruf – extern

Member JCL803

Legen Sie die in der Aufgabe 8.1 definierte Prozedur in der Datei userid.KURS.JCL als Member PRC001 an.
Erstellen Sie einen Job, der diese Prozedur aufruft.

Lassen Sie den Job laufen und prüfen Sie die Ausgabe und das Ergebnis.

Vorgehensweise

Editieren userid.KURS.JCL(JCL803); kopieren Member JCL801. Daraus die Zeilen der Prozedur in Member PRC001 auslagern. Prozedurbibliothek definieren und Prozedurnamen anpassen (Prozedurname = Membername!!).

```
//RZSRGEN JOB (000947,TEST,,,,9900),
//        'ZIT P 2.21,SEIDLER',MSGCLASS=T,CLASS=G,
//        NOTIFY=RZSR,MSGLEVEL=(1,1)
//*** AUFRUF EXTERNE PROZEDUR PRC001
//PROZLIB  JCLLIB ORDER=&SYSUID..KURS.JCL
//GO01     EXEC PROC=PRC001
```

8.4 Prozeduraufruf – extern, parametrisiert

Member: JCL804

Erstellen Sie einen Job analog der Aufgabe 0, der die Prozedur PRC002 aus der Datei userid.KURS.JCL aufruft. Bereiten Sie die Prozedur entsprechend vor.
Lassen Sie den Job laufen und prüfen Sie die Ausgabe und das Ergebnis.

Vorgehensweise

Editieren userid.KURS.JCL(JCL804); kopieren Member JCL802. Daraus die Zeilen der Prozedur in Member PRC002 auslagern. Prozedurbibliothek definieren und Prozedurnamen anpassen (Prozedurname = Membername!!).

```
//RZSRGEN JOB (000947,TEST,,,,9900),
//        'ZIT P 2.21,SEIDLER',MSGCLASS=T,CLASS=G,
//        NOTIFY=RZSR,MSGLEVEL=(1,1)
//*** AUFRUF EXTERNE PROZEDUR PRC002
//PROZLIB  JCLLIB ORDER=&SYSUID..KURS.JCL
//GO01     EXEC PROC=PRC002,DATEI=&SYSUID..TEST.PS
```

8.5 Prozeduraufruf – MSGLEVEL

Testen Sie die Aufgaben 8.1, 0, 8.3 und 8.4 mit unterschiedlichen Angaben für MSGLEVEL: Prüfen Sie die Unterschiede der Ergebnisse.

Geben Sie auch ungültige Werte an und sehen Sie sich die Jobinformationen dazu an.