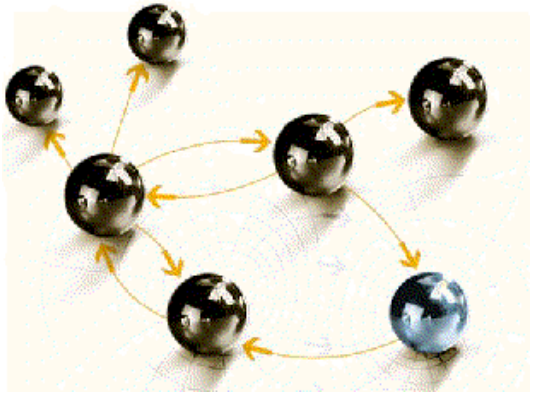


cps4it

z/OS Job Control Language

Grundlagen

cps4it
consulting, projektmanagement und seminare für die informationstechnologie
Ralf Seidler, Stromberger Straße 36A, 55411 Bingen
Fon: +49-6721-992611, Fax: +49-6721-992613, Mail: ralf.seidler@cps4it.de
Internet: http://www.cps4it.de

Ziele des Seminars

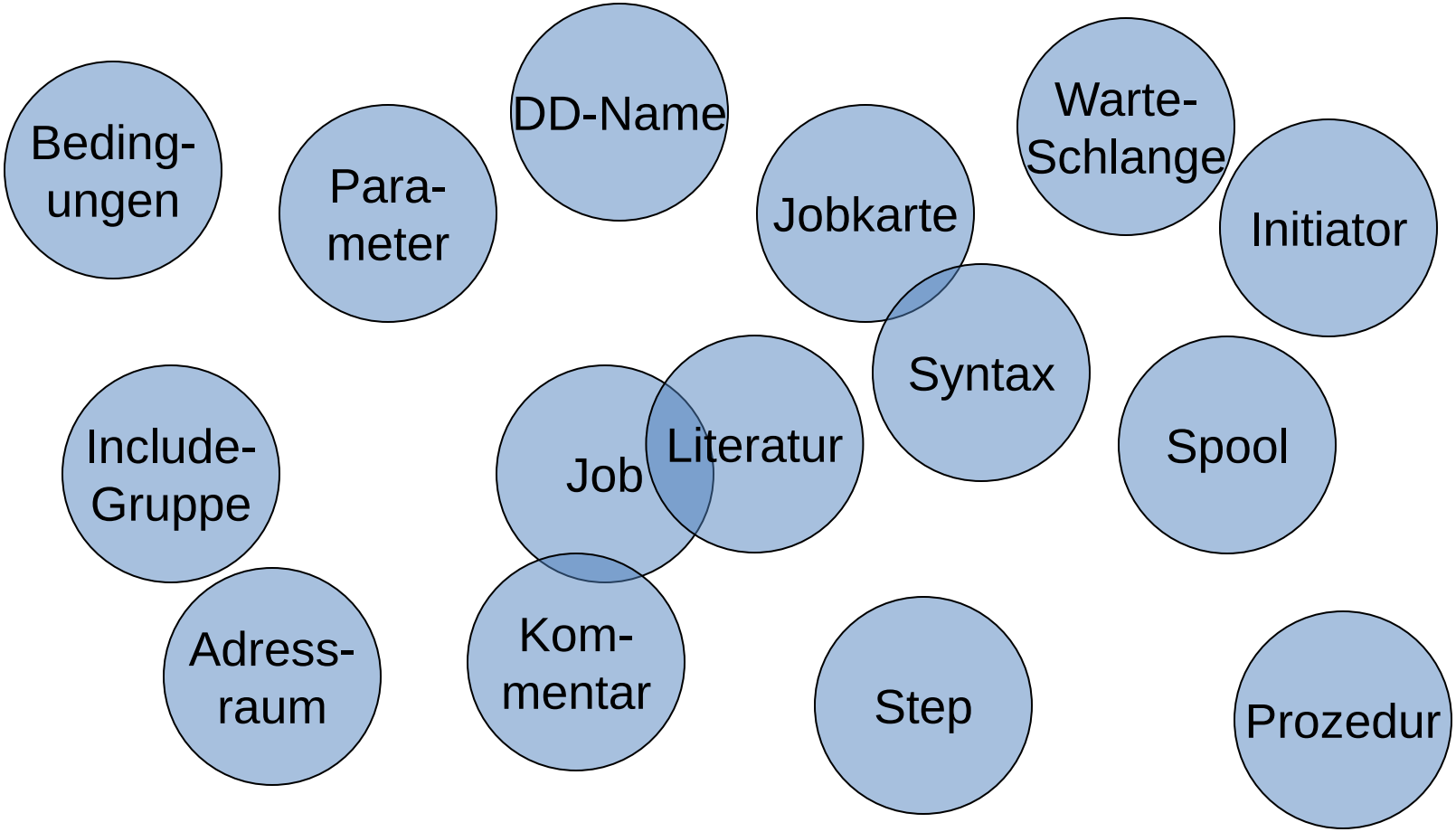
- Sprache JCL kennen lernen
- Syntax der JCL beherrschen
- fit in Datei-Formen werden
- Praxisbeispiele kennen lernen
- üben … üben … üben
- Besonderheiten

Inhalt

- Einführung
- Job-Beschreibung, Step-Beschreibung
- Datei-Beschreibung (1)
- Datei-Beschreibung (2)
- Standard- und Dienstprogramme - Überblick
- Job-Steuerung, Step-Steuerung
- Datei-Beschreibung (3)
- Include-Gruppe, JCL-Prozedur
- Diskussion und Austausch

Einführung

Begriffe



Einführung

Literatur

- IBM Dokumentation z.B.
  - https://www-01.ibm.com/servers/resourcelink/svc00100.nsf/pages/zOSV2R3MvsPublications?OpenDocument
  - https://www.ibm.com/support/knowledgecenter/en/SSLTBW_2.3.0/com.ibm.zos.v2r3.iea/iea.htm
  - https://www.ibm.com/support/knowledgecenter/en/SSLTBW_2.3.0/com.ibm.zos.v2r3.has/has.htm
  - https://www.ibm.com/support/knowledgecenter/en/SSLTBW_2.3.0/com.ibm.zos.v2r3.iat/iat.htm
- Internetseiten siehe Suchmaschinen z.B.
  - http://www.isc.ucsb.edu/tsg/jcl.html
  - http://www.mainframes.com/JCL.html
- G.D.Brown:

  JCL – Jobcontrol Language im Betriebssystem z/OS

  4. Auflage

  ISBN 978-3486273977

  (gebraucht!!!)

~~64,80€~~

~~69,95€~~

> 98 €

Einführung

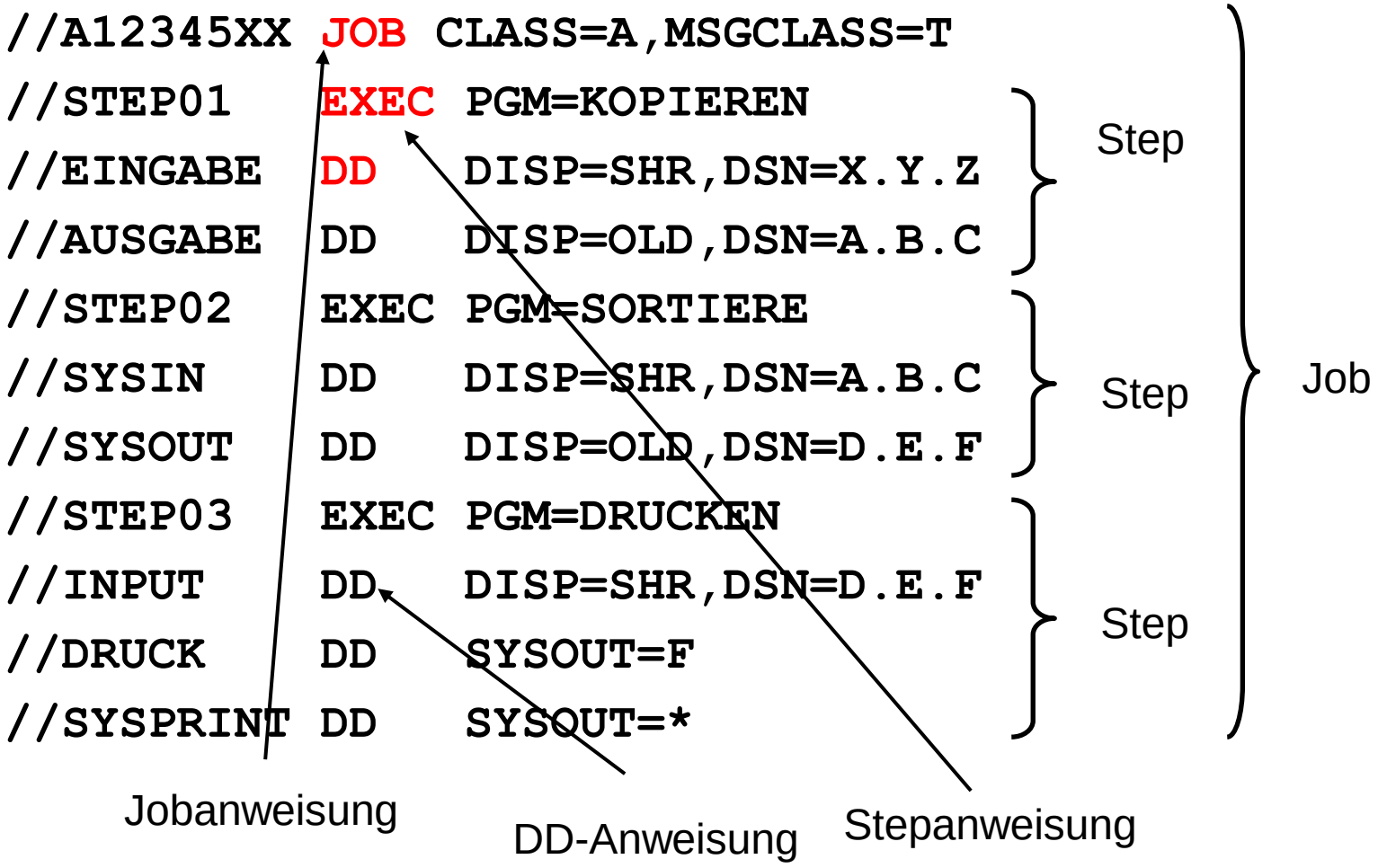
JCL heißt …

- Jakarta Commons Logging
- JEDI Code Library
- Johann Christian Lotter
- Jazzclub Lippstadt
- Jar Class Loader
- Johannes C. Laxander

- Job Control Language

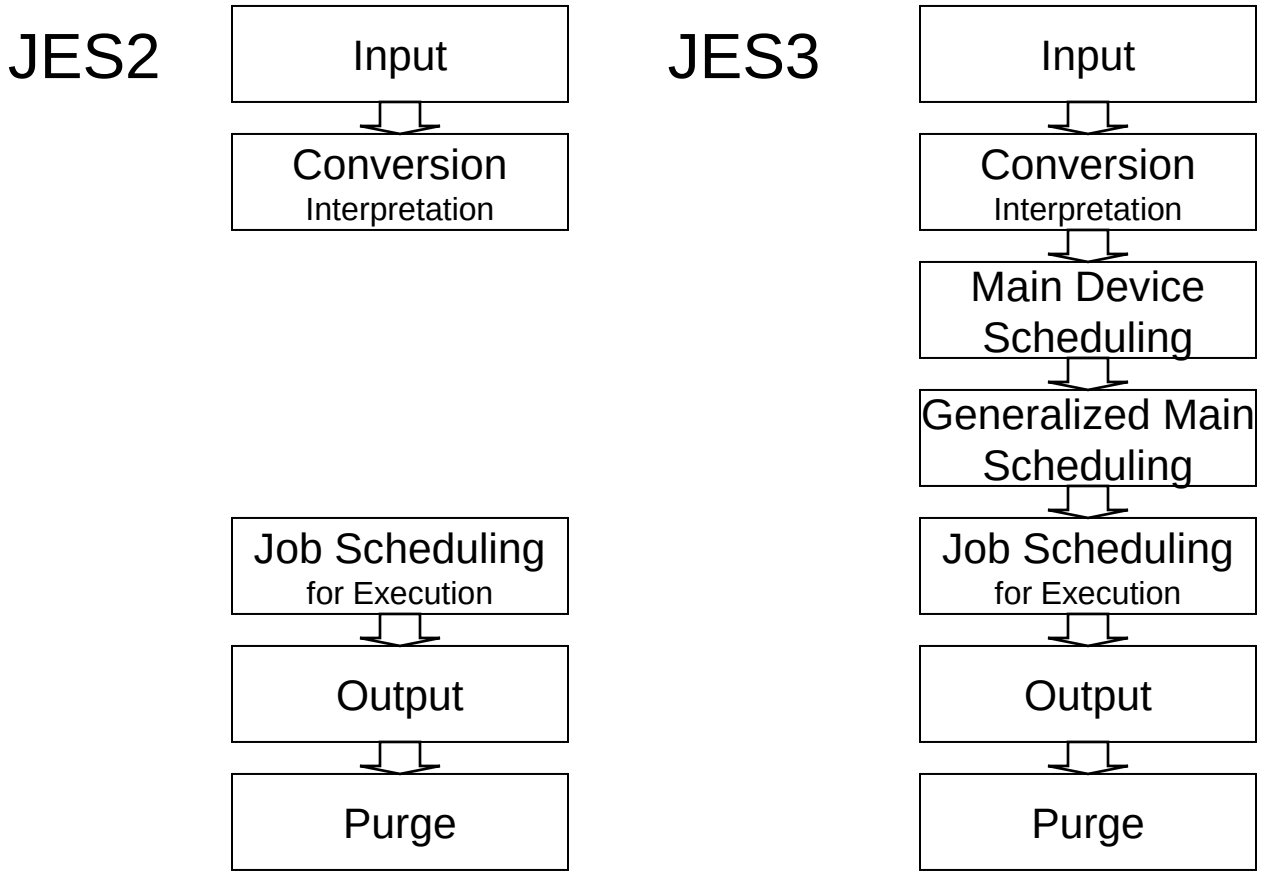
Einführung

Job Control Language



Einführung

JES2 / JES3



Einführung

Eingabe-, Ausgabe-Warteschlangen / Adressräume

Warteschlangen

A	B	C	D	E	. . .	Z	0	. . .	9
J009									
J004									
J003		J008							
J001	J002	J005				J006	J007		

Adressräume

		J009 JP:E	J004 JP:A DA:F DA:Z		J010 JP:E DA:E	J008 JP:A DA:L	
MAS- TER	SYS- TEM	INIT1 C=AC	INIT2 C=A	INIT3 C=C	INIT4 C=9ZE	INIT5 C=D	TSO

Ausgabe-Warteschlangen

A	B	C	D	E	. . .	Z	0	. . .	9
J003 JP									
J005 JP						J003 DA			
J001 JP DA		J003 DA		J006 JP		J005 DA			

Einführung

CLASS / Scheduling Environment

- Moderne Umsetzung von CLASS= und Definition der Initiator
- Definition in den Unternehmen verschieden
- CLASS beschreibt Priorität / Laufzeit wann / Laufzeit wie lange
- SCHENV sagt aus, welche Subsysteme benötigt werden

Einführung

JCL-Anweisungen – Überblick

Name	Bezeichnung	Anweisung
//jobname	JOB	Job-Anweisung
//stepname	EXEC	EXEC-Anweisung
//ddname	DD	DD-Anweisung
//*		Kommentar-Anweisung
/*		Delimiter-Anweisung
//name	IF/THEN/ELSE/ENDIF	IF/THEN . . . -Anweisung
//name	JCLLIB	JCL-Library-Anweisung
//name	INCLUDE	INCLUDE-Anweisung
//procname	PROC	PROC-Anweisung
//name	PEND	PEND-Anweisung
//name	SET	SET-Anweisung
//outname	OUTPUT	OUTPUT-Anweisung
//		Jobende

Einführung

JCL-Anweisungen – Syntax – 1

```
//name  op-code  [param1][,param2]... [kommentar]
|1 |3      |12         |17                    |71 |72
```

Bezeichnung	Spalte	Inhalt	Spalte - Erläuterung
Identifikationsfeld	1 - 2	//	genau da
Namensfeld	3 - 10	Name	genau ab 3
Operationsfeld	12 - 15	Operation Code	bis 15 anfangen
Parameterfeld	17 - 71	Parameter	ab 4, spätestens ab 17
Kommentarfeld	4 - 71	Kommentare	
Fortsetzungsfeld	72	Fortsetzungszeichen	

Einführung

JCL-Anweisungen – Syntax – 2

```
//[name]  op-code  [param1][,param2]...                    [num]
                                                          |  |
                                                          |72|
```

```
//[name]  op-code  [param1][,param2-1]...                  *  [num]
//                       [param2-2][,param3]...                [num]
  |  |                                                        |  |
  |  |                                                        |  |
  4  16                                                      72
```

Einführung

JCL-Anweisungen - Parameterarten

- Positionsparameter
  - //XV10733A JOB RUV,SEIDLER
  - //XV10733A JOB ,SEIDLER
- Schlüsselwortparameter
  - //XV10733A JOB ...,CLASS=A,MSGCLASS=Y
- Subparameter
  - //ddname DD DSN=datei.a.b,
// DISP=(NEW,CATLG,DELETE)
- Positions- und Schlüsselwortparameter
  - //XV10733A JOB ,SEIDLER,CLASS=A,MSGCLASS=Y

Einführung

JCL-Anweisungen – Parameterarten – Beispiel

```
//XV8822DA JOB '1N28900301020010000',
//            'MIG-TEAM',
//            TIME=(,30),
//            MSGLEVEL=(1,1),
//            MSGCLASS=Q,
//            CLASS=APPSHORT,
//            SCHENV='PRD_DBALL',
//            NOTIFY=&SYSUID    ,REGION=100M
//*MAIN ROOM=AL21-L2X,LINES=5000,CLASS=APPSHORT
//*-------------------------------------*
```

Einführung

Übung(en)

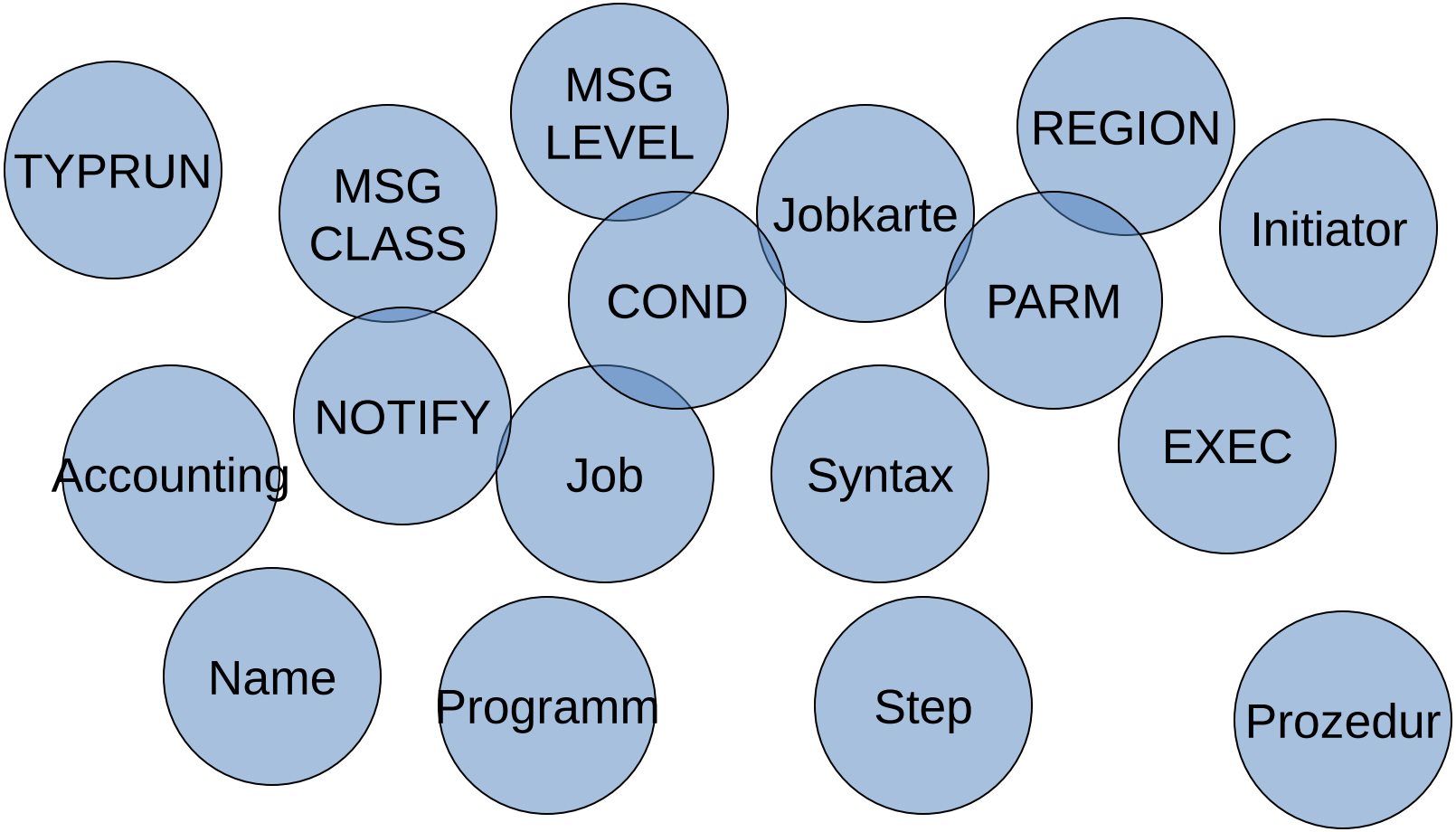
- Kapitel 1.1: Auswahl und Test User-ID
- Kapitel 1.2: Bibliothek erstellen

Inhalt

- Einführung
- Job-Beschreibung, Step-Beschreibung
- Datei-Beschreibung (1)
- Datei-Beschreibung (2)
- Standard- und Dienstprogramme - Überblick
- Job-Steuerung, Step-Steuerung
- Datei-Beschreibung (3)
- Include-Gruppe, JCL-Prozedur
- Diskussion und Austausch

Job-Beschreibung, Step-Beschreibung

Begriffe



Job-Beschreibung, Step-Beschreibung

JOB-Anweisung

- ist erste Anweisung
- definiert Beginn eines Jobs
- es gibt
  - Namensfeld
  - Operationsfeld
  - Parameterfeld

- Standards beachten

Job-Beschreibung, Step-Beschreibung

JOB-Anweisung – Abrechnungsinformation

- Syntax
  - ([accounting-number][,accounting-information]...)
    - abhängig von Verrechnungskriterien wie
      - Speicherplatz
      - I/O-Einheiten
      - CPU
      - Hauptspeicherleistung

- Beispiel:

```
//JOBX  JOB  (3SLX510,000,00T NR0003)
//JOB1  JOB  (CPS4IT,TRAINING)
//JOB2  JOB  'CPS4IT,TRAINING'
```

Job-Beschreibung, Step-Beschreibung

JOB-Anweisung – Programmierername

- Syntax
  - [,programmierer-name]
    optional

- Beispiel:

```
//JOB1  JOB  (CPS4IT,TRAINING),'R.SEIDLER'
//JOB2  JOB  ,'SEIDLER'
//JOB3  JOB (CPS4IT,TRAINING),'MAYER&&HUBER'
```

Job-Beschreibung, Step-Beschreibung

Übung(en)

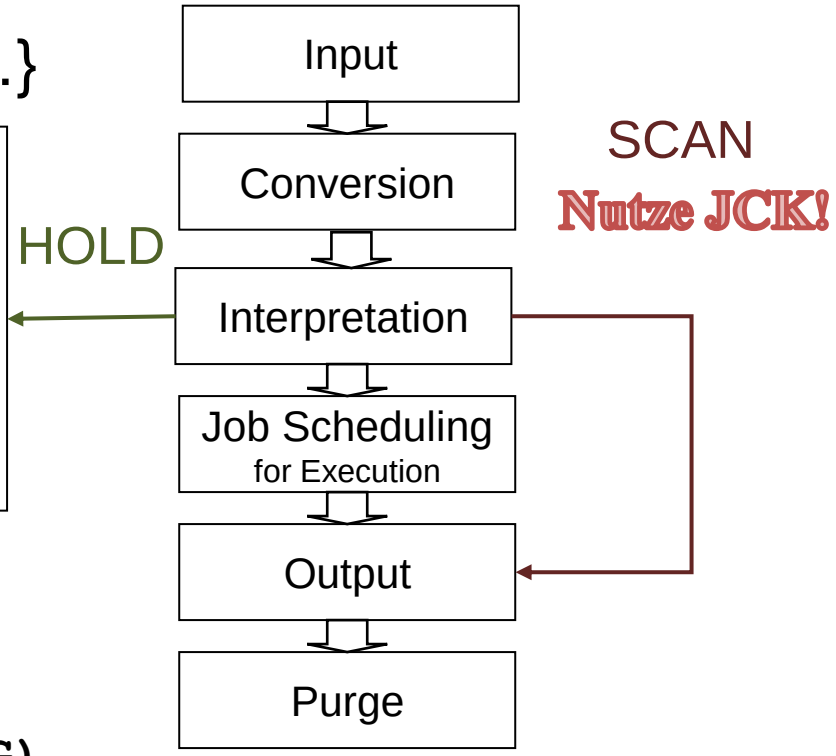
- Kapitel 2.1: minimale Jobkarte erstellen
  - Jobkarte erstellen
  - Submit
  - Ausgabe analysieren

Job-Beschreibung, Step-Beschreibung

JOB-Anweisung – TYPRUN

- Syntax
  - TYPRUN={HOLD|SCAN|…} optional

QUEUE	CLASS
JOB1	C
JOB2	E
JOB3	C
JOB4	H
JOB5	E



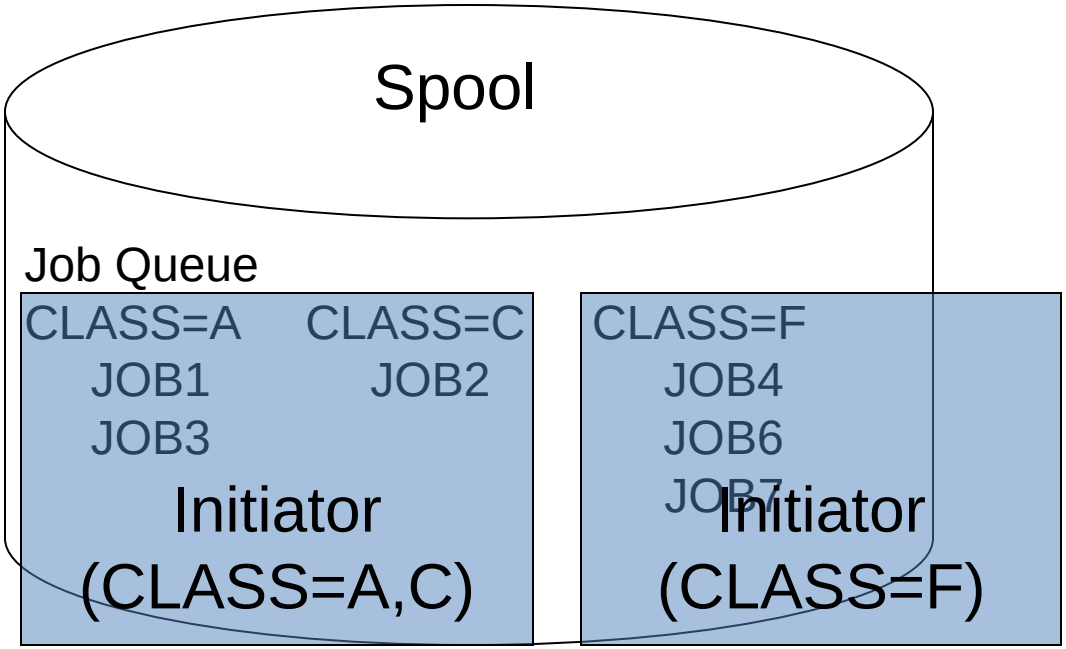
- Beispiel:

```
//JOB1   JOB   (CPS4IT,TRAINING),
//             'R.SEIDLER',TYPRUN=SCAN
```

Job-Beschreibung, Step-Beschreibung

JOB-Anweisung – CLASS

- Syntax
  - CLASS=jobclass optional
  - Nutze Konvention im Unternehmen



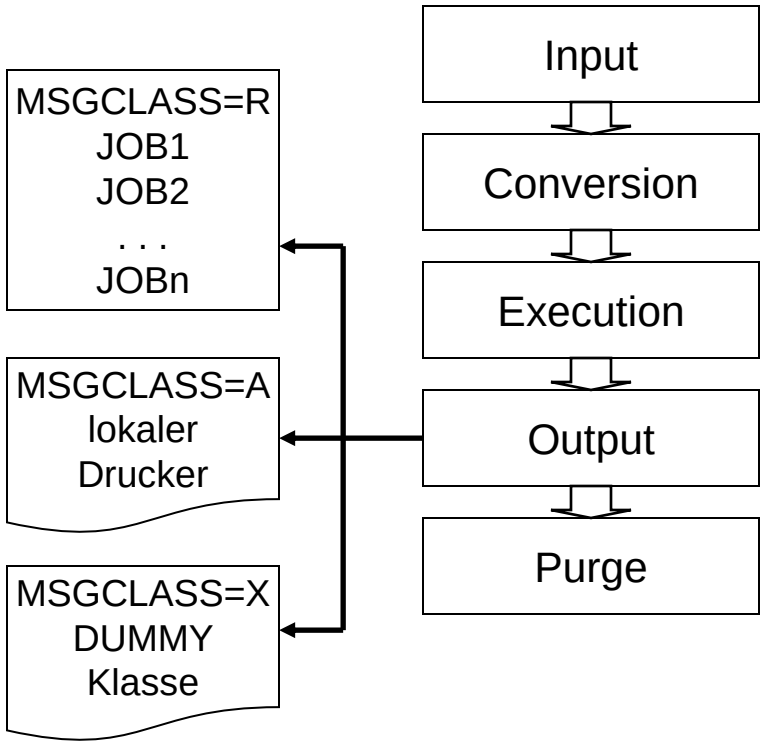
- Beispiel:

```
//JOB1   JOB   ,(CPS4IT,TRAINING),CLASS=APPSHORT
//JOB2   JOB   ,(CPS4IT,TRAINING),CLASS=APPLATE
```

Job-Beschreibung, Step-Beschreibung

JOB-Anweisung – MSGCLASS

- Syntax
  - MSGCLASS=class optional



- Beispiel:

```
//JOBX  JOB  (CPS4IT,TRAINING),SEIDLER,
//           CLASS=APPSHORT,MSGCLASS=R
```

Job-Beschreibung, Step-Beschreibung

JOB-Anweisung – MSGLEVEL

- Syntax
  - MSGLEVEL=([statements][,messages])
    - statements € {0,1,2}
    - messages € {0,1}
      (Die unterschiedlichen Auswirkungen sehen wir uns während der Übungen an.)

- Beispiel:

```
//JOBX  JOB  ,SEIDLER,CLASS=APPSHORT,MSGLEVEL=(1,1)
```

Job-Beschreibung, Step-Beschreibung

Übung(en)

- Kapitel 2.2: Jobkarte erweitern
  - CLASS
  - MSGCLASS
  - verschiedene Angaben für die Parameter testen

Job-Beschreibung, Step-Beschreibung

JOB-Anweisung – TIME

- Syntax
  - TIME=([minutes][,seconds])
- Hinweis: es gibt Konventionen auf Basis CLASS= und SCHENV=
- TIME=NOLIMIT ist nicht erwünscht

- Beispiel:

```
//JOB1   JOB   ,SEIDLER,CLASS=A,TIME=(1,30)
//JOB2   JOB   ,SEIDLER,CLASS=A,TIME=(,30)
//JOB3   JOB   ,SEIDLER,CLASS=C,TIME=1
//JOB4   JOB   ,SEIDLER,CLASS=K,TIME=NOLIMIT
//JOB5   JOB   ,SEIDLER,CLASS=X,TIME=1440
```

Job-Beschreibung, Step-Beschreibung

JOB-Anweisung – REGION

- Syntax
  - REGION= { nnnnnnK / mmmmM }

Hinweise:

- **Region<=16M -> below; sollte nicht mehr benutzt werden**
- **Region=200M ist eine gute Wahl, wenn nötig**

- Beispiel:

```
//JOB1   JOB   ,SEIDLER,CLASS=A,REGION=4096K
//JOB2   JOB   ,SEIDLER,CLASS=C,REGION=4M
//JOB3   JOB   ,SEIDLER,CLASS=C,REGION=0M
```

Job-Beschreibung, Step-Beschreibung

JOB-Anweisung – COND

- Syntax
  - COND=([returncode][,operator])

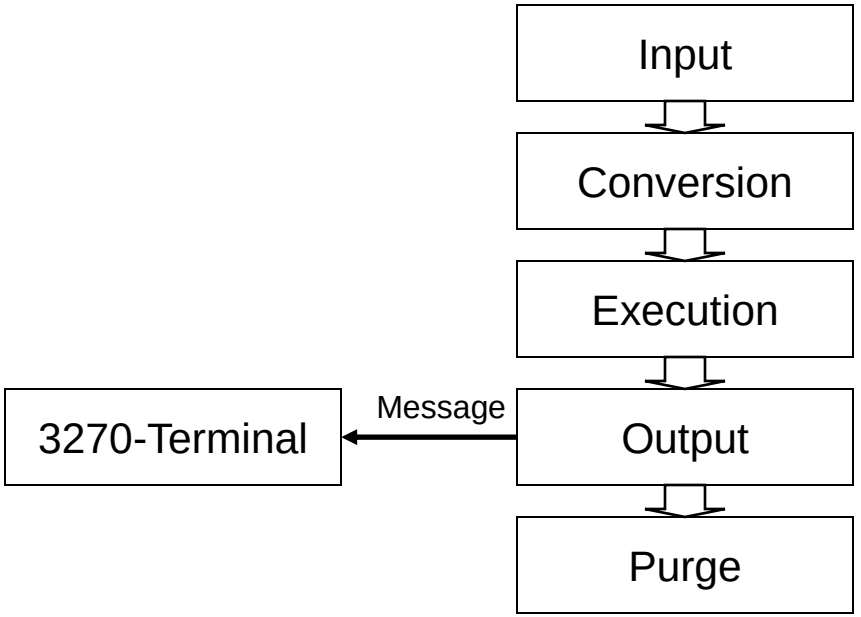
- Beispiel:

```
//JOBX  JOB  ,SEIDLER,CLASS=C,COND=(0,NE)
//JOBY  JOB  ,SEIDLER,CLASS=C,COND=(8,LT)
```

Job-Beschreibung, Step-Beschreibung

JOB-Anweisung – NOTIFY

- Syntax
  - NOTIFY=userid



- Beispiel:

```
//JOB1   JOB   ,SEIDLER,CLASS=A,COND=(0,NE),
//             NOTIFY=XV10733
//JOB2   JOB   ,SEIDLER,CLASS=C,COND=(8,LT),
//             NOTIFY=&SYSUID
```

Job-Beschreibung, Step-Beschreibung

JOB-Anweisung – BYTES, LINES, PAGES

- Syntax
  - BYTES=zahl1
  - LINES=zahl2
  - PAGES=zahl3

- Beispiel:

```
//JOB1   JOB   ,SEIDLER,CLASS=A,COND=(0,NE),
//             BYTES=1000
//JOB2   JOB   ,SEIDLER,CLASS=C,COND=(8,LT),
//             LINES=100,PAGES=1000
```

Job-Beschreibung, Step-Beschreibung

Übung(en)

- Kapitel 2.3: Jobkarte erweitern
  - NOTIFY
  - REGION
  - TIME

Job-Beschreibung, Step-Beschreibung

EXEC-Anweisung – Überblick – 1

- definiert Beginn eines Steps
- ruft Programm oder Prozedur auf
- beendet vorherigen Step
- es gibt
  - Namensfeld
  - Operationsfeld
  - Parameterfeld

Job-Beschreibung, Step-Beschreibung

EXEC-Anweisung – Überblick – 2

- Syntax
  - //[stepname] EXEC [pos-par][,schl-par]

- Beispiel:

```
//JOB1  JOB  ,SEIDLER,CLASS=X,MSGCLASS=X
//STEP1 EXEC   PGM=IEBGENER
      . . .
//STEP2 EXEC   PGM=IEHLIST
      . . .
//STEP3 EXEC   PROC=SORT
      . . .
```

Job-Beschreibung, Step-Beschreibung

EXEC-Anweisung – PGM

- Syntax
  - PGM=programm-name

- Beispiel:

```
//JOB1   JOB   ,SEIDLER,CLASS=X,MSGCLASS=X
//STEP1  EXEC    PGM=IEBGENER
       . . .
//STEP2  EXEC    PGM=MYSORT
//STEPLIB  DD   DSN=userlib,DISP=SHR
```

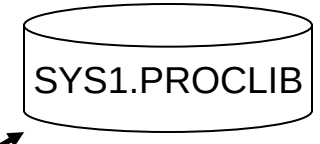
Job-Beschreibung, Step-Beschreibung

EXEC-Anweisung – PROC

- Syntax
  - [PROC=]prozedur-name

- Beispiel:

```
//JOB1  JOB  ,SEIDLER,CLASS=X,MSGCLASS=X
//STEP1 EXEC   PROC=DRUCKEN
      . . .
//STEP2 EXEC   KOPIEREN
```



Job-Beschreibung, Step-Beschreibung

Übung(en)

- Kapitel 2.4: Jobstep erstellen
- Kapitel 2.5: einen 2. Step hinzufügen

Job-Beschreibung, Step-Beschreibung

EXEC-Anweisung – PARM

- Syntax
  - PARM=information

- Beispiel:

```
//JOBX  JOB  ,SEIDLER,CLASS=X,MSGCLASS=X
//STEP1 EXEC   PGM=MYPROC,PARM='TEST'
```

Job-Beschreibung, Step-Beschreibung

EXEC-Anweisung – TIME

- Syntax
  - TIME=([minutes][,seconds])

- Beispiel:

```
//STEP1 EXEC   PGM=PGM01,TIME=(1,30)
//STEP2 EXEC   PGM=PGM02,TIME=(,15)
//STEP3 EXEC   PGM=PGM03,TIME=NOLIMIT
```

Job-Beschreibung, Step-Beschreibung

EXEC-Anweisung – REGION

- Syntax
  - REGION= { nnnnnnK / mmmmM }

Hinweise:
- **Region<=16M -> below; sollte nicht mehr benutzt werden**
- **Region=200M ist eine gute Wahl, wenn nötig**

- Beispiel:

```
//JOBX  JOB  ,SEIDLER,CLASS=X,MSGCLASS=X
//STEP1 EXEC PGM=MYPROG,REGION=1024K
```

Job-Beschreibung, Step-Beschreibung

EXEC-Anweisung – COND

- Syntax
  - COND= { (code,op[,stepname]) | EVEN | ONLY | ((code,op[,stepname]),(code,op[,stepname])]...[,EVEN]) | ((code,op[,stepname]),(code,op[,stepname])]...[,ONLY]) }

- Detaillierung später

Job-Beschreibung, Step-Beschreibung

Übung(en)

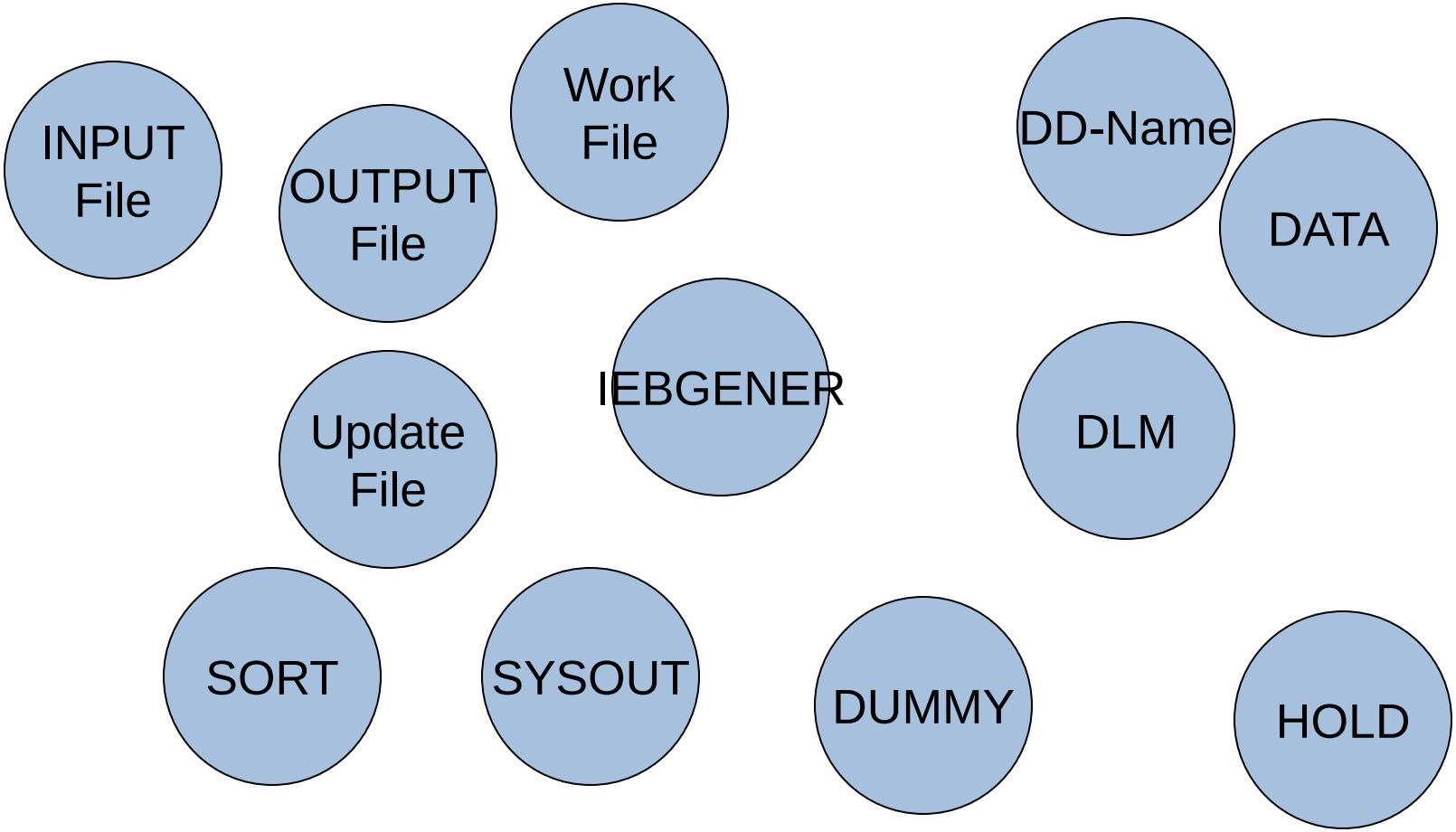
- Kapitel 2.6: den 2. Step abhängig von COND laufen lassen bzw. nicht laufen lassen

Inhalt

- Einführung
- Job-Beschreibung, Step-Beschreibung
- Datei-Beschreibung (1)
- Datei-Beschreibung (2)
- Standard- und Dienstprogramme - Überblick
- Job-Steuerung, Step-Steuerung
- Datei-Beschreibung (3)
- Include-Gruppe, JCL-Prozedur
- Diskussion und Austausch

Datei-Beschreibung (1)

Begriffe



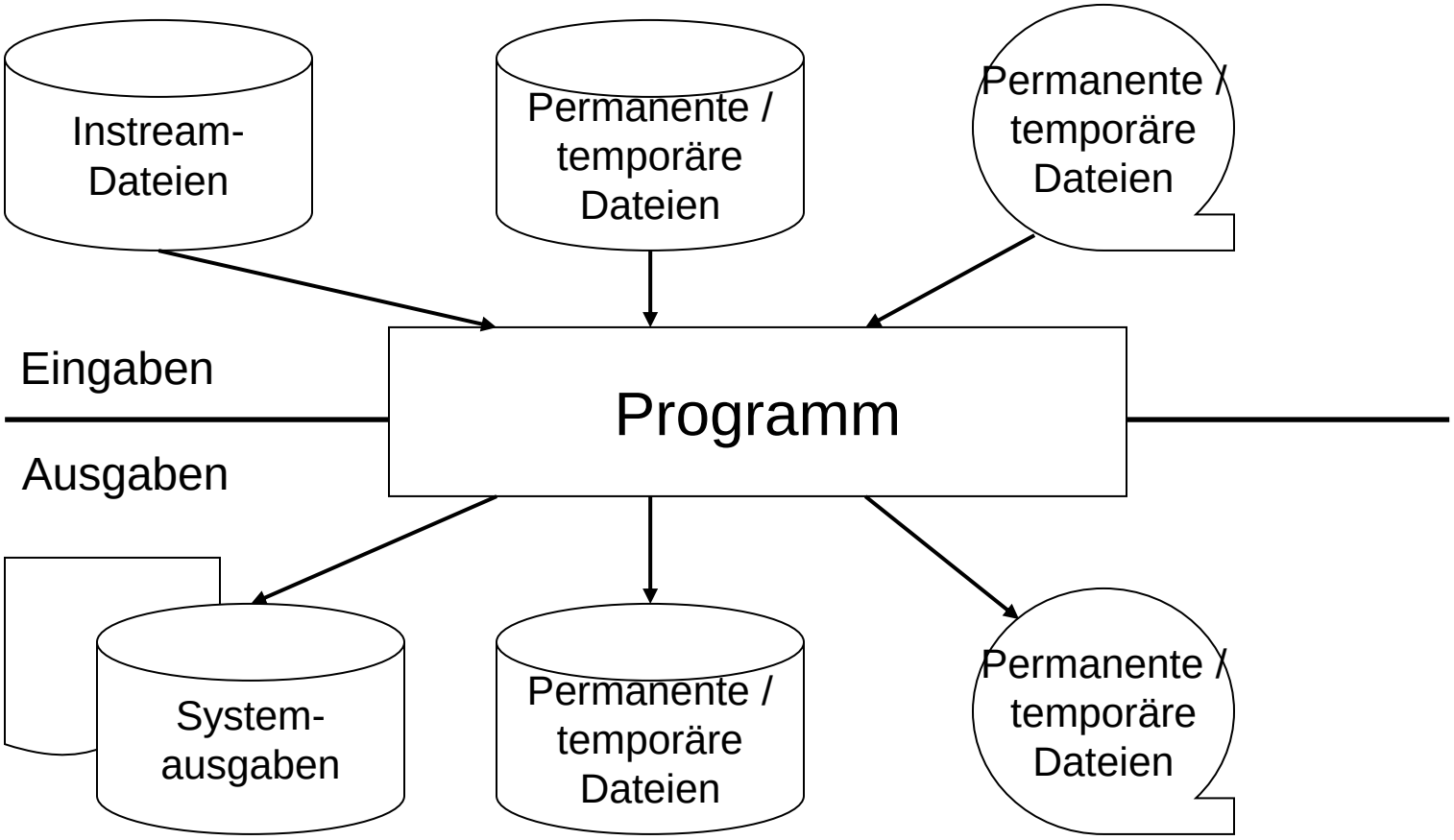
Datei-Beschreibung (1)

DD-Anweisung (1) – Überblick – 1

- Verwendung bei Step / Programm
- Art der Verwendung
  - Input / Output / Update / Work
- Art des Datenträgers
  - Drucker / Band / Platte / optische Platten
- Lebensdauer
  - permanent / temporär / Systemeingabe / Systemausgabe

Datei-Beschreibung (1)

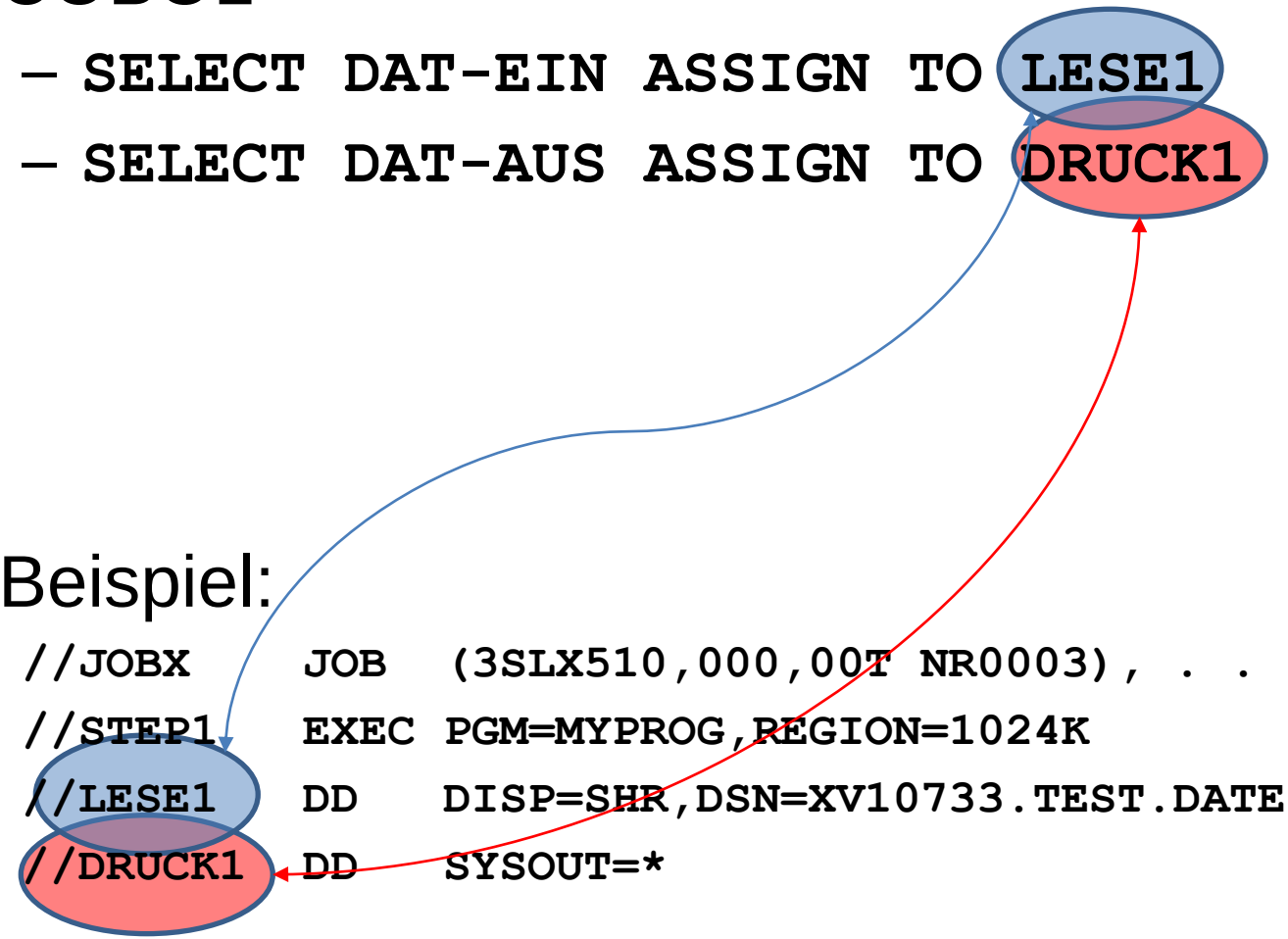
DD-Anweisung (1) – Überblick – 2



Datei-Beschreibung (1)

DD-Anweisung (1) – Bezug zum Programm – 1

- COBOL
  - **SELECT DAT-EIN ASSIGN TO LESE1**
  - **SELECT DAT-AUS ASSIGN TO DRUCK1**



- Beispiel:

```
//JOBX    JOB  (3SLX510,000,00T NR0003), . . .
//STEP1   EXEC PGM=MYPROG,REGION=1024K
//LESE1   DD   DISP=SHR,DSN=XV10733.TEST.DATEI
//DRUCK1  DD   SYSOUT=*
```

Datei-Beschreibung (1)

DD-Anweisung (1) – Bezug zum Programm – 2

- COBOL

```
– 03  sam1-ddname          pic  x(08)  value  'DDSE01'.
– 03  sam1-request-type    pic  x(06)  value  'OPEN  '.
– 03  sam1-file-type       pic  x(01)  value  'I'.

– 03  sam2-ddname          pic  x(08)  value  'DDSA01'.
– 03  sam2-request-type    pic  x(06)  value  'OPEN  '.
– 03  sam2-file-type       pic  x(01)  value  'O'.
```

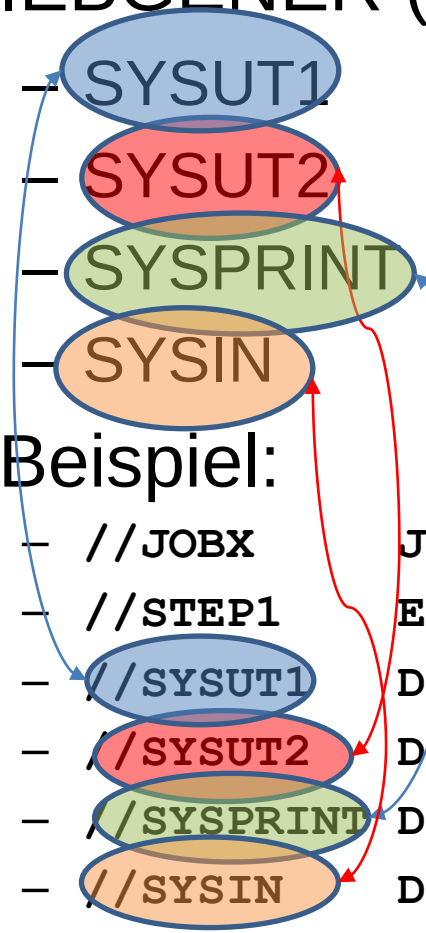
- Beispiel:

```
//JOBX    JOB  (3SLX510,000,00T NR0003), . . .
//STEP1   EXEC PGM=MYPROG,REGION=1024K
//DDSE01  DD   DISP=SHR,DSN=XV10733.TEST.DATEI(+0)
//DDSA01  DD   DISP=(NEW..),DSN=XV10733.TEST.DATEI(+1)
```

Datei-Beschreibung (1)

DD-Anweisung (1) – Bezug zum Programm – 3

- IEBGENER (und Derivate) – einfache Zuordnung



  - SYSUT1 Eingabe-Datei
  - SYSUT2 Ausgabe-Datei
  - SYSPRINT Protokoll
  - SYSIN Steueranweisung

- Beispiel:

```
//JOBX     JOB  (3SLX510,000,00T NR0003), . . .
//STEP1    EXEC PGM=MYPROG,REGION=1024K
//SYSUT1   DD DUMMY
//SYSUT2   DD SYSOUT=*
//SYSPRINT DD SYSOUT=*
//SYSIN    DD DUMMY
```

Datei-Beschreibung (1)

DD-Anweisung – Felder

- Syntax
  - //ddname DD [pos-par][schl-par]... [kommentar]

- Beispiel:

```
//JOBX     JOB  (3SLX510,000,00T NR0003), . . .
//STEP01   EXEC PGM=ZINS
//EINGABE  DD *
00011981198219831984 1985
00021986198719881989 1990
00031991199219931994 1995
//AUSGABE  DD SYSOUT=*
```

Datei-Beschreibung (1)

System-Eingabe – DATA

- Syntax
  - //ddname DD { * | DATA } [par]... [kommentar]
- Beispiel:

```
//JOBX     JOB  (3SLX510,000,00T NR0003), . . .
//STEP01   EXEC PGM=ZINS
//EINGABE  DD *
000119811982198319841985
000219861987198819891990
000319911992199319941995
//AUSGABE  DD SYSOUT=A
```

Datei-Beschreibung (1)

DD-Anweisung – DLM – 1 (JES2)

- Syntax
  - DLM=delimiter

- Beispiel (JES2):

```
//JOBX     JOB  (3SLX510,000,00T NR0003), . . .
//STEP01   EXEC PGM=ZINS
//EINGABE  DD *,DLM='??'
00011981198219831984 1985
00021986198719881989 1990
00031991199219931994 1995
//AUSGABE  DD SYSOUT=A
??
//SYSPRINT   DD DSN=XV10733.ispf.datei,DISP=SHR
```

Datei-Beschreibung (1)

DD-Anweisung – DLM – 1 (JES3)

- Syntax
  - DLM=delimiter

- Beispiel (JES3):

```
//JOBX     JOB  (3SLX510,000,00T NR0003), . . .
//STEP01   EXEC PGM=ZINS
//EINGABE  DD *,DLM='??'
00011981198219831984 1985
00021986198719881989 1990
00031991199219931994 1995
//AUSGABE  DD SYSOUT=A
??
//SYSPRINT   DD DSN=XV10733.ispf.datei,DISP=SHR
```

Datei-Beschreibung (1)

DD-Anweisung – DLM – 2

- Beispiel 2:

```
//JOBX      JOB  (3SLX510,000,00T NR0003), . . .
//STEP01    EXEC PGM=ZINS
//EINGABE   DD DATA,DLM='&&&&'
00031991199219931994 1995
//AUSGABE   DD SYSOUT=A
&&
//SYSPRINT    DD DISP=SHR,DSN=XV10733.ispf.datei
```

Datei-Beschreibung (1)

DD-Anweisung – DLM – 3

DD-Anweisung	JES2	JES3	Bemerkung
DD *	~~/*~~ oder //	~~/*~~ oder //	Normalfall
DD *,DLM='xx'	xx oder //	xx	vermeiden!!
~~DD DATA~~	~~/*~~	~~/*~~	
~~DD DATA,DLM='xx'~~	~~xx~~	~~xx~~	~~Nur notwendig in JES2, wenn // in Daten~~

Datei-Beschreibung (1)

DD-Anweisung – SYSOUT, HOLD

- Syntax
  - SYSOUT= { class | ([class][,INTRDR][,form-name]) | * }

- Beispiel:

```
//JOBX     JOB  (3SLX510,000,00T NR0003), . . .
//STEP01   EXEC PGM=ZINS
//EINGABE  DD ...
//SYSPRINT DD SYSOUT=*
//OUTSTAT  DD SYSOUT=(V,,R#01)
//SYSOUT   DD SYSOUT=(,INTRDR)
//OUTSYS   DD SYSOUT=F,HOLD=YES
```

Datei-Beschreibung (1)

DD-Anweisung – SYSOUT, HOLD

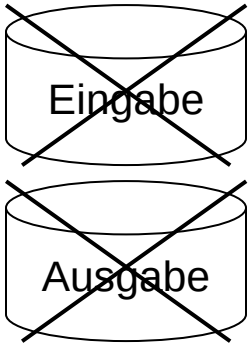
- Syntax
  - COPIES= { nnn | (nnn,(grp-wert[,grp-wert]...)) }
- Beispiel:

```
//SYSPRINT DD SYSOUT=V,COPIES=3
```

Datei-Beschreibung (1)

DD-Anweisung – SYSOUT, HOLD

- Syntax
  - //ddname DD DUMMY[parameter]
    - //EINGABE DD DUMMY
    - //AUSGABE DD DUMMY



- Beispiel:

```
//JOBX     JOB  (3SLX510,000,00T NR0003), . . .
//STEP01   EXEC PGM=ZINS
//EINGABE  DD ...
//SYSPRINT DD DUMMY
```

Datei-Beschreibung (1)

Einfache Anwendungen – IEBGENER

- Syntax (symbolisch)
  - //stepname EXEC PGM=IEBGENER
  - //SYSUT1 - Eingabe
  - //SYSUT2 - Ausgabe
  - //SYSPRINT - Protokoll
  - //SYSIN - Steueranweisung

Datei-Beschreibung (1)

Übung(en)

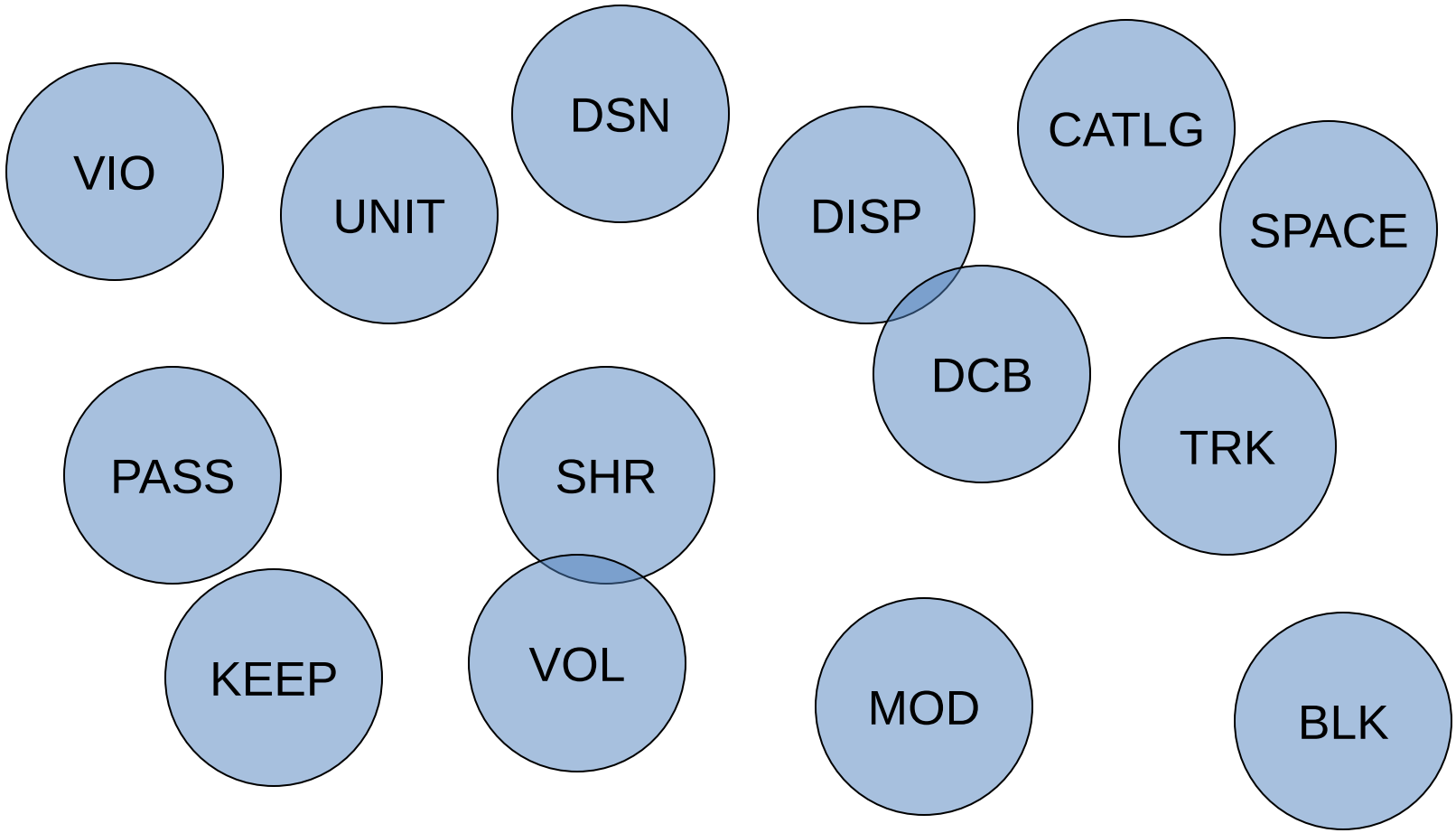
- Kapitel 3.1: Lesen einer Instream-Datei
- Kapitel 3.2: Lesen einer Dummy-Datei
- Kapitel 3.3: Schreiben in das “Nirwana”

Inhalt

- Einführung
- Job-Beschreibung, Step-Beschreibung
- Datei-Beschreibung (1)
- Datei-Beschreibung (2)
- Standard- und Dienstprogramme - Überblick
- Job-Steuerung, Step-Steuerung
- Datei-Beschreibung (3)
- Include-Gruppe, JCL-Prozedur
- Diskussion und Austausch

Datei-Beschreibung (2)

Begriffe



Datei-Beschreibung (2)

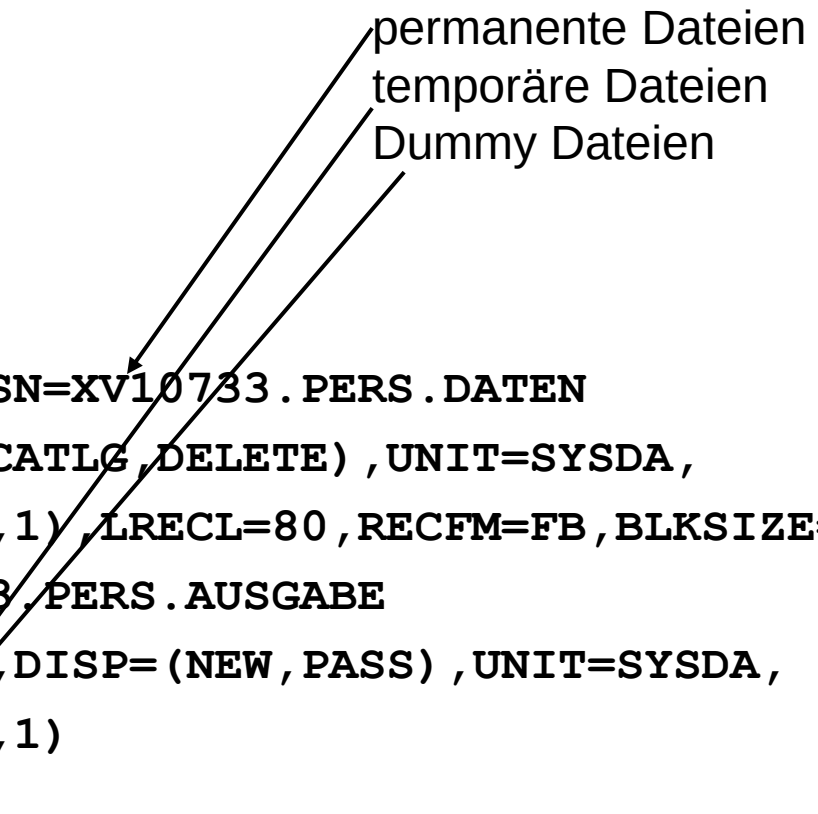
DD-Anweisung (2) – Überblick

- bisher:
  - Systemausgabe
  - Systemeingabe
  - Dummy
- neu:
  - permanente Dateien auf Platte oder Band
  - temporäre Dateien auf Platte oder Band
  - verschiedene Organisationsformen (VSAM, seq.)

Datei-Beschreibung (2)

DD-Anweisung – DSNAME

- Syntax
  - DSN[AME]=dateiname



- Beispiel:

```
//SYSIN    DD DISP=SHR,DSN=XV10733.PERS.DATEN
//SYSPRINT DD DISP=(NEW,CATLG,DELETE),UNIT=SYSDA,
//            SPACE=(TRK,1),LRECL=80,RECFM=FB,BLKSIZE=0,
//            DSN=XV10733.PERS.AUSGABE
//WORKFILE DD DSN=&&TEMP,DISP=(NEW,PASS),UNIT=SYSDA,
//            SPACE=(TRK,1)
//NOCHWAS  DD DUMMY
```

Datei-Beschreibung (2)

Datenbestände – DISP

- Syntax
  - DISP=status
  - DISP=([status][,normal-end][,abnormal-end])

status vor Stepbeginn	normal-end normales Stepende	abnormal-end Stepabbruch
NEW	,DELETE	,DELETE
OLD	,KEEP	,KEEP
SHR	,CATLG	,CATLG
MOD	,UNCATLG	,UNCATLG
	,PASS	

- Anmerkungen zur Folgeseite:

  Fall 1: Die Datei existiert bereits bei Jobbeginn oder wird während des Jobs mit KEEP oder CATLG behandelt.
  Fall 2: Die Datei existiert nicht bei Jobbeginn und wird während des Jobs nicht mit KEEP oder CATLG behandelt.

Datei-Beschreibung (2)

Datenbestände – DISP – Tabelle

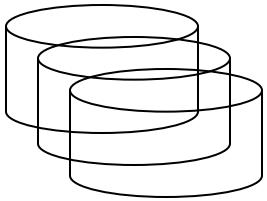
Überblick DISP-Parameter						
1. Subparm	2. Subparm	3.Subparm	Normales Stepende	Abnormales Stepende	Jobende (kein Abend)	Jobende (nach Abend)
NEW, MOD als NEW	<weglassen>	<weglassen>	DELETE	DELETE		
NEW, MOD als NEW	KEEP oder DELETE oder CATLG	<weglassen>	<wie 2. Subparm>	<wie 2. Subparm>		
NEW, MOD als NEW	KEEP oder DELETE oder CATLG	KEEP oder DELETE oder CATLG	<wie 2. Subparm>	<wie 3. Subparm>		
NEW, MOD als NEW	PASS	<weglassen>	PASS	PASS	DELETE	DELETE
NEW, MOD als NEW	PASS	KEEP oder DELETE oder CATLG	PASS	PASS		<wie 3. Subparm>
NEW, (für temp)	<weglassen>	<beliebig>	DELETE	DELETE		
NEW, (für temp)	DELETE	<beliebig>	DELETE	DELETE		1. Fall: KEEP 2. Fall: KEEP
NEW, (für temp)	PASS	<beliebig>	PASS	DELETE	DELETE	1. Fall: KEEP 2. Fall: KEEP
OLD,SHR MOD als OLD	<weglassen>	<weglassen>	1. Fall: KEEP 2. Fall: Delete	1. Fall: KEEP 2. Fall: Delete		
OLD,SHR MOD als OLD	KEEP oder DELETE oder CATLG oder UNCATLG	<weglassen>	<wie 2. Subparm>	<wie 2. Subparm>		
OLD,SHR MOD als OLD	KEEP oder DELETE oder CATLG oder UNCATLG	KEEP oder DELETE oder CATLG oder UNCATLG	<wie 2. Subparm>	<wie 3. Subparm>		
OLD,SHR MOD als OLD	PASS	<weglassen>	PASS	PASS	1. Fall: KEEP 2. Fall: Delete	1. Fall: KEEP 2. Fall: Delete
OLD,SHR MOD als OLD	PASS	KEEP oder DELETE oder CATLG oder UNCATLG	PASS	PASS	1. Fall: KEEP 2. Fall: Delete	<wie 3. Subparm>

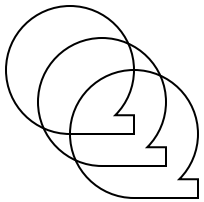
Datei-Beschreibung (2)

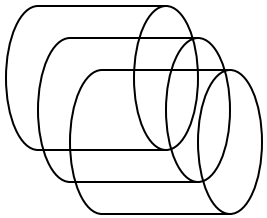
Datenbestände – UNIT / VOL

- Syntax
  - UNIT=device-type|group-name
  - VOL[UME]=SER=vol-nummer

SYSDA REEL TAPE etc.







Datei-Beschreibung (2)

Datenbestände – SPACE (ohne SMS)

- Syntax

  SPACE=({ blklen, | TRK, | CYL, } (prim[,sec][,dir])[,RLSE])

- Beispiel:

```
//SYSIN    DD DISP=SHR,DSN=XV10733.PERS.DATEN
//SYSPRINT DD DISP=(NEW,CATLG,DELETE),UNIT=SYSDA,
//            SPACE=(TRK,1),LRECL=80,RECFM=FB,BLKSIZE=0,
//            DSN=XV10733.PERS.AUSGABE
//WORKFILE DD DSN=&&TEMP,DISP=(NEW,PASS),UNIT=SYSDA,
//            SPACE=(TRK,1)
```

siehe ISPF 3.2

Datei-Beschreibung (2)

Datenbestände – Eigenschaften

- Syntax
  - DCB=(subparm[,subparm] . . .)
  - Data Control Block

- Beispiel:

```
//AUSGABE  DD DISP=(NEW,CATLG,DELETE),UNIT=SYSDA,
//            SPACE=(TRK,1),DSN=XV10733.PERS.AUSGABE,
//            DCB=(LRECL=80,RECFM=FB,BLKSIZE=0)    oder
//            LRECL=80,RECFM=FB,BLKSIZE=0
```

Datei-Beschreibung (2)

Datenbestände – DCB-Bezüge

- Reihenfolge ist zu beachten
  - aus Programm
  - aus DD-Anweisung
  - aus Definition in SMS
  - ~~aus Definition der Datei auf Platte/Band~~

- Anmerkung:
  - Es gibt keine Faustregel, was wo angegeben werden soll; eine genaue Kenntnis der Anwendung ist erforderlich.
  - Sinnvoll: so wenig Angaben wie möglich machen.

Datei-Beschreibung (2)

Datenbestände – LABEL

- Syntax
  - LABEL=[seq-nummer][,labeltyp]
  - labeltype=SL|NSL|NL|BLP

Datei-Beschreibung (2)

Datenbestände – EXPDT, RETPD

- Syntax
  - EXPDT=yyddd oder
  - EXPDT=yyyy/ddd

  - RETPD=nnnn

Datei-Beschreibung (2)

Datenbestände – Katalogeintrag

Katalogeinträge				
DSN	UNIT	VOLSER	LABEL	sonst.
A.B.C	3390	ABC111	-	-
A.B.D	3390	ABC111	-	-
KLM.NOP.QRS	3380	KXX202	-	-
KLM.NOP.QRS.BACKUP	3480	T32145	1	-
RST.UVW.XYZ	3480	T28282	1	-

Datei-Beschreibung (2)

cps4it

Übung(en)

- Kapitel 4.1: Anlegen PS-Datei mit IEFBR14
- Kapitel 4.2: Anlegen PO-Datei
- Kapitel 4.3: Kopieren Datei in PO-Member
- Kapitel 4.4: Kopieren PO-Member

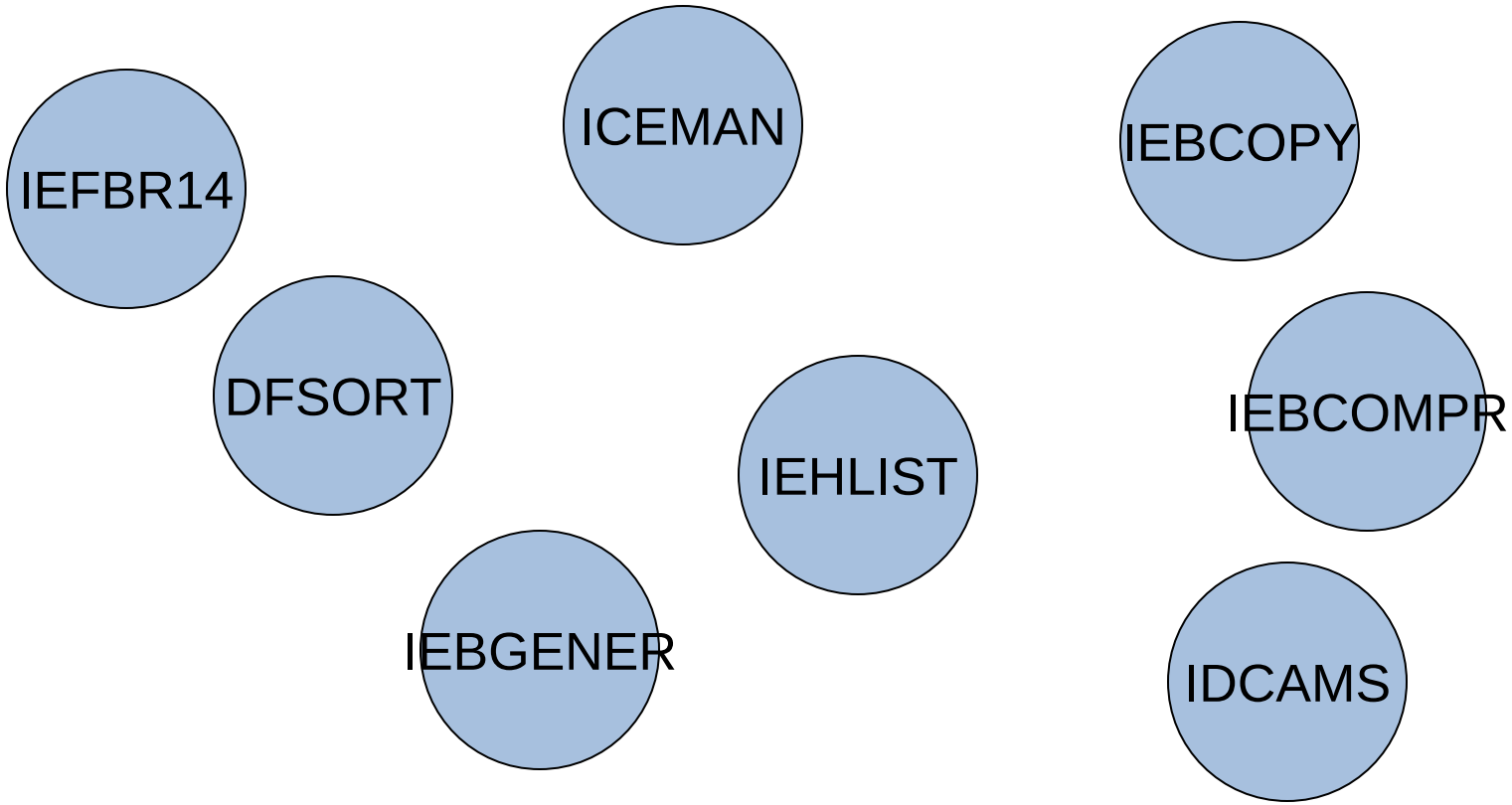
- jeweils
  - Job wegschicken
  - Output analysieren

Inhalt

- Einführung
- Job-Beschreibung, Step-Beschreibung
- Datei-Beschreibung (1)
- Datei-Beschreibung (2)
- Standard- und Dienstprogramme - Überblick
- Job-Steuerung, Step-Steuerung
- Datei-Beschreibung (3)
- Include-Gruppe, JCL-Prozedur
- Diskussion und Austausch

Standard- und Dienstprogramme

Begriffe



Datei-Beschreibung (2)

Übung(en)

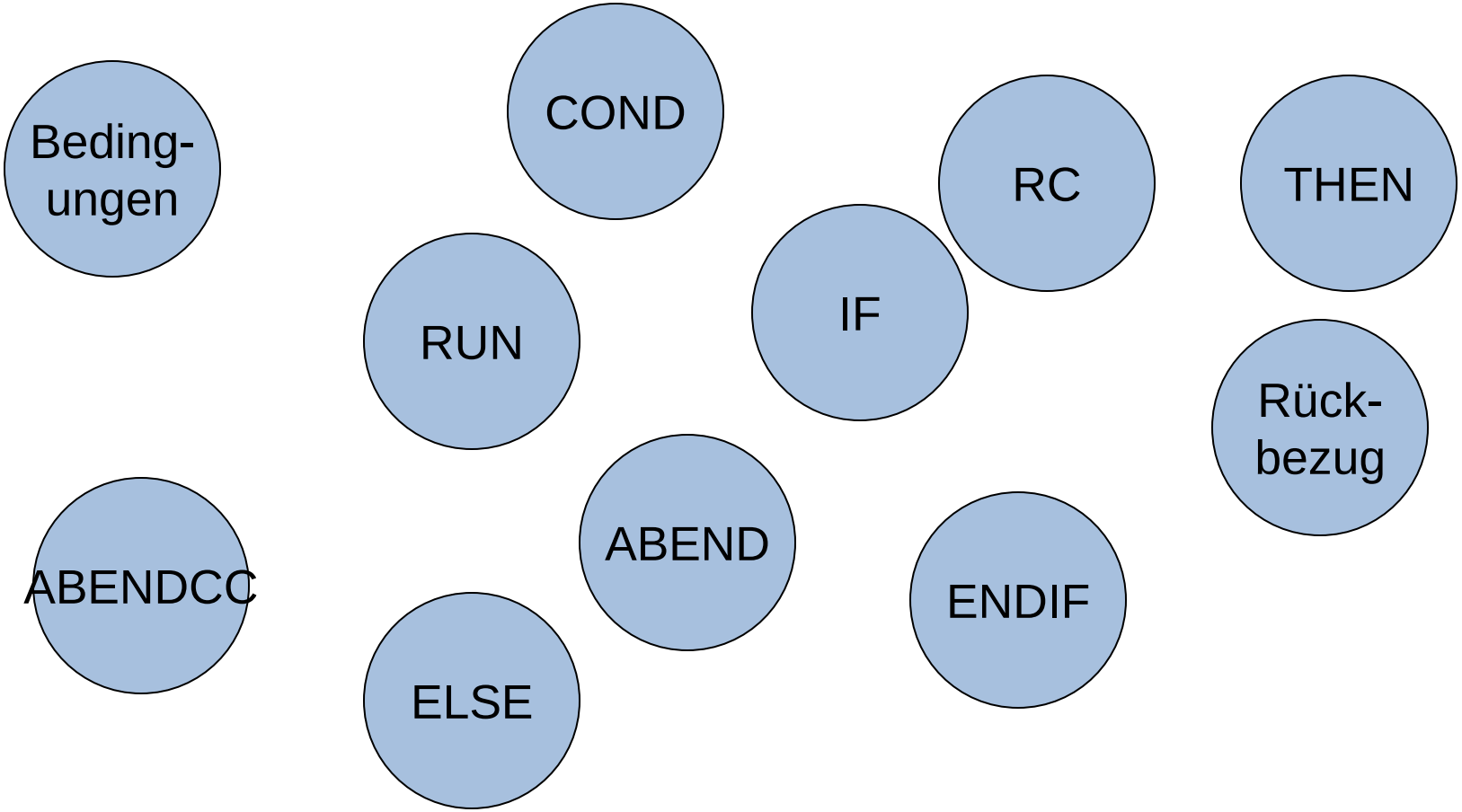
- Kapitel 5.1: Anlegen DSN mit IEFBR14
- Kapitel 5.2: Editieren Datei im EDIT
- Kapitel 5.3: Kopieren Datei mit IEBGENER
- Kapitel 5.4: Löschen DSN mit IEFBR14

Inhalt

- Einführung
- Job-Beschreibung, Step-Beschreibung
- Datei-Beschreibung (1)
- Datei-Beschreibung (2)
- Standard- und Dienstprogramme - Überblick
- Job-Steuerung, Step-Steuerung
- Datei-Beschreibung (3)
- Include-Gruppe, JCL-Prozedur
- Diskussion und Austausch

Job-Steuerung, Step-Steuerung

Begriffe



Job-Steuerung, Step-Steuerung

Allgemeines – Überblick

- Ziel
  - Jobablauf kontrollieren
  - Stepablauf kontrollieren
  - keine manuelle Eingriffe erforderlich
- Methoden
  - Bedingungen für vorzeitiges Ende eines Jobs
  - Bedingungen für Unterdrücken eines Steps
  - Bedingungen für Unterdrücken oder Ausführens von Steps oder Stepfolgen

Job-Steuerung, Step-Steuerung

Allgemeines – Regeln für Stepsteuerung

- Grundregeln
  - maximal 255 Steps pro Job
  - Reihenfolge ist top-down
  - bei Abend wird alles Folgende unterdrückt
- Formelles Ende eines Steps
  - normales Ende, Programm liefert Return-Code
  - abnormales Ende, ABEND-Code Sxxx|Unnnn
  - Step wird unterdrückt (NOT EXECUTED oder FLUSH)

Job-Steuerung, Step-Steuerung

COND-Parameter – JOB

- Syntax
  - COND=(zahl,operator)
  - COND=((zahl,operator), (zahl,operator),...)

- Beispiel:

```
//JOBX      JOB   ,SEIDLER,CLASS=A,COND=(0,NE)
//JOBY      JOB   ,SEIDLER,CLASS=A,COND=(4,LE)
//JOBX      JOB   ,SEIDLER,CLASS=A,COND=((8,EQ),(12,EQ))
```

Job-Steuerung, Step-Steuerung

COND-Parameter – EXEC

- Syntax
  - COND=(zahl,operator,[step])
  - COND={EVEN|ONLY}
  - COND=((zahl,operator,[step]),..., {EVEN|ONLY})

- Beispiel:

```
//STEP01   EXEC PGM=PROG01
//STEP02   EXEC PGM=PROG02,COND=(0,NE)
//STEP03   EXEC PGM=PROG03,
//         COND=((4,LT,STEP01),(8,LT,STEP02),EVEN)
```

Job-Steuerung, Step-Steuerung

COND-Parameter – EXEC – Syntaxerklärung

- COND=(zahl,operator,[step]):
  wenn Bedingung erfüllt ist, wird Step geskipped
- COND=EVEN:
  Step ausführen, auch wenn (even) ein Step einen Abbruch hatte
- COND=ONLY:
  Step ausführen, nur wenn vorher ein Abbruch stattfand
- https://www.ibm.com/support/knowledgecenter/SSLTBW_2.1.0/com.ibm.zos.v2r1.ieab600/iea3b6_Examples_of_the_COND_parameter.htm

Job-Steuerung, Step-Steuerung

COND-Parameter – EXEC – Beispiel mit Erklärung – 1

```
//STEP6  EXEC PGM=DISKUTIL,COND=(4,GT,STEP3)
```

if the return code from STEP3 is 0 through 3, the system bypasses STEP6. If the return code is 4 or greater, the system executes STEP6. Because neither EVEN nor ONLY is specified, the system does not execute this step if a preceding step abnormally terminates.

Job-Steuerung, Step-Steuerung

COND-Parameter – EXEC – Beispiel mit Erklärung – 2

```
//STEP6  EXEC PGM=DISKUTIL,COND=(4,GT,STEP3)
```

Falls RC von STEP3 0,1,2,3 ist, wird Step6 geskipped. Ist er 4 oder größer wird Step6 ausgeführt. Da weder EVEN noch ONLY vorhanden sind, wird der Step nicht ausgeführt, wenn vorher ein Abbruch war.

Job-Steuerung, Step-Steuerung

COND-Parameter – EXEC – Beispiel mit Erklärung – 3a

```
//TEST2  EXEC PGM=DUMPINT,
//            COND=((16,GE),(90,LE,STEP1),ONLY)
```

Der Step wird nur (ONLY) dann ausgeführt, wenn die folgenden Bedingungen erfüllt sind:

Ein vorheriger Step hatte einen Abbruch.
Kein RC-Test ist erfüllt.

. . .

Job-Steuerung, Step-Steuerung

COND-Parameter – EXEC – Beispiel mit Erklärung – 3b

```
//TEST2  EXEC PGM=DUMPINT,
//            COND=((16,GE),(90,LE,STEP1),ONLY)
```

Der Step wird also dann ausgeführt, wenn alle drei folgenden Bedingungen erfüllt sind:

- Ein vorheriger Step hatte einen Abbruch.
- Die RC aller vorheriger Steps sind >=17.
- Der RC von STEP1 ist <=89

Job-Steuerung, Step-Steuerung

COND-Parameter – EXEC – Beispiel mit Erklärung – 3c

```
//TEST2  EXEC PGM=DUMPINT,
//            COND=((16,GE),(90,LE,STEP1),ONLY)
```

Der Step wird also dann nicht ausgeführt, wenn nur eine der folgenden Bedingungen erfüllt sind:

- Alle vorherigen Steps sind normal beendet.
- Die RCs aller vorherigen Steps sind 0 bis 16.
- Der RC von STEP1 ist >= 90.

Job-Steuerung, Step-Steuerung

COND-Parameter – Tabelle 1

Ausführen oder Unterdrücken des aktuellen Steps abhängig vom COND-Parameter		
Eintragung im COND-Parameter	Returncode (RC) eines früheren Steps	
	aktuellen Step ausführen?	aktuellen Step unterdrücken?
COND=(code,GT)	RC >= code	RC < code
COND=(code,GE)	RC > code	RC <= code
COND=(code,EQ)	RC ^= code	RC = code
COND=(code,LT)	RC <= code	RC > code
COND=(code,LE)	RC < code	RC >= code
COND=(code,NE)	RC = code	RC ^= code

Job-Steuerung, Step-Steuerung

COND-Parameter – Tabelle 2

Wirkung von EVEN / ONLY auf die Ausführung eines Steps			
	ABEND vorher?	RC-Test erfüllt?	Step ausführen?
EVEN	nein	nein	ja
EVEN	nein	ja	nein
EVEN	ja	nein	ja
EVEN	ja	ja	nein
ONLY	nein	nein	nein
ONLY	nein	ja	nein
ONLY	ja	nein	ja
ONLY	ja	ja	nein
keiner	nein	nein	ja
keiner	nein	ja	nein
keiner	ja	nein	nein
keiner	ja	ja	nein

Job-Steuerung, Step-Steuerung

IF/THEN, ELSE, ENDIF – Überblick

- dient zur Steuerung der Steps oder Stepfolgen eines Jobs
- Logik wie in Programmiersprachen
- 15 Stufen erlaubt

Job-Steuerung, Step-Steuerung

IF/THEN, ELSE, ENDIF – Syntax

- Syntax
  - //[name] IF bedingung THEN
  - ja-Zweig
  - [//[name] ELSE
  - nein-Zweig]
  - //[name] ENDIF

Job-Steuerung, Step-Steuerung

IF/THEN, ELSE, ENDIF – Syntax Bedingungen

- Bedingungen
  - [step.]RC operator zahl
  - [step.]ABENDCC operator zahl
  - [step.]ABEND operator T/F
  - [step.]RUN operator T/F

GT	>
LT	<
NG	^>
NL	^<

EQ	=
NE	^=
GE	>=
LE	<=

NOT	^
AND	&
OR	|

Job-Steuerung, Step-Steuerung

IF/THEN, ELSE, ENDIF – RC

- Test auf höchsten bisherigen Returncode
- Test auf Returncode eines Steps
- Syntax: RC op code oder step.RC op code

0 <= code < 4096

- Beispiel:

```
//STEP01   EXEC PGM=PROG01
             .   .   .
//IFTEST   IF (RC <= 8) THEN
//STEP02   EXEC PGM=PROG02 etc.
//IFTEST   ELSE
//STEP03   EXEC PGM=PROG03 etc.
//IFTEST   ENDIF
//STEP04   EXEC PGM=PROG04
```

GT	>
LT	<
NG	^>
NL	^<

EQ	=
NE	^=
GE	>=
LE	<=

Job-Steuerung, Step-Steuerung

IF/THEN, ELSE, ENDIF – ABENDCC

- Test auf zuletzt aufgetretenen ABEND-Code
- Test auf ABEND-Code eines Steps
- Syntax
  - ABENDCC op code oder step.ABENDCC op code
- code ist
  - System-ABEND-Code
    - Sxxx mit xxx = {x’001’, ... , x’FFF’}
  - User-ABEND-Code
    - Unnnn mit nnnn = {0001, ... ,4095}

EQ	=
NE	^=

Job-Steuerung, Step-Steuerung

IF/THEN, ELSE, ENDIF – ABEND

- Test auf irgendeinen aufgetretenen ABEND
- Test auf ABEND eines Steps
- Syntax
  - ABEND[=TRUE] oder step.ABEND[=TRUE]
  - ABEND=FALSE oder step.ABEND=FALSE
  - NOT ABEND oder NOT step.ABEND
  - ^ABEND oder ^step.ABEND

Job-Steuerung, Step-Steuerung

IF/THEN, ELSE, ENDIF – RUN

- Test, ob bestimmter Step ausgeführt wurde

- Syntax
  - step.RUN=TRUE oder step.RUN
  - step.RUN=FALSE oder NOT step.RUN oder ^step.RUN

Job-Steuerung, Step-Steuerung

IF/THEN, ELSE, ENDIF – sonstiges

- Unterdrückte Steps werden bei Tests nicht berücksichtigt (außer bei RUN)
- COND bei EXEC wird innerhalb IF geprüft
- manche Systemabends führen zu generellem Abbruch des Jobs
- IF vor 1. Step: Auswertung **nach** 1. Step!
- Prioritätenfolge:
  - Klammer, NOT, AND, OR
  - von links nach rechts

Bezug / Rückbezug – Überblick

- Verwendung
  - dsname
    - Datei ist katalogisiert
  - *.ddname
    - ddname im gleichen Step
  - *.step.ddname
    - ddname eines anderen Steps
  - *.step.procstep.ddname
    - ddname eines Steps in einer Prozedur

Job-Steuerung, Step-Steuerung

Bezug / Rückbezug – Bezugnamen

- bei DD-Anweisungen
  - VOLUME
    - VOL=REF=dsname
  - DCB-Parameter
    - DCB=dsname
    - zusätzliche Parameter überschreiben
  - LIKE-Parameter (nur SMS / viiiiel besser)
    - LIKE=dsname (später)

- bei DD-Anweisungen
  - DSN-Parameter
  - VOLUME
    - VOL=REF=dsname
  - DCB-Parameter
    - DCB=dsname
    - zusätzliche Parameter überschreiben
  - REFDD-Parameter (nur SMS)
  - OUTPUT-Parameter (später)

Job-Steuerung, Step-Steuerung

Bezug / Rückbezug – Rückbezugnamen – 2

- Syntax
  - DSNAME=*[.step].ddname

- Beispiel:

```
//COB2   EXEC PGM=IGYCRCTL,PARM='OBJECT'
//SYSLIN DD DSN=&&OBJLIB(MEM1),DISP=(,PASS),
//          UNIT=SYSDA,SPACE=(TRK,(5,2))
            . . .
//LKED   EXEC PGM=HEWL,PARM='XREF,MAP'
//SYSLIN DD DSN=*.COB2.SYSLIN,DISP=(OLD,DELETE)
   oder
//SYSLIN DD DSN=&&OBJLIB(MEM1),DISP=(OLD,DELETE)
```

Job-Steuerung, Step-Steuerung

Übung(en)

- Kapitel 6.1a: Anlegen PS-Datei-1
- Kapitel 6.1b: Anlegen PS-Datei-2 mit Rückbezug
- Kapitel 6.2: Kopieren Datei mit eventueller Neuanlage

Job-Steuerung, Step-Steuerung

MAIN-Statement in JES3

- Syntax
  - //*MAIN parameter[,parameter]
- Parameter in Auswahl
  - CLASS=class-name
  - DEADLINE={(time,type[,date])}
    {(time,type[,rel,cycle])}
    - Hinweis: In Jobkarte ist die Angabe PRTY=0 erforderlich
  - siehe JCL-Reference Kapitel JES3
- Beispiel:

```
//*MAIN CLASS=WLM23
//*MAIN DEADLINE=(1225,A,12/31/2014)
```

Job-Steuerung, Step-Steuerung

SCHEDULE-Statement in JES2

- Syntax
  - //[name] SCHEDULE parameter[,parameter]
- Parameter in Auswahl
  - AFTER= / BEFORE= / DELAY= / STARTBY=
  - Siehe MVS JCL Reference V2R3 Kapitel 27

```
STARTBY= {'+HH:MM'}
({[ 'HH:MM' ] , [ 'MM/DD/YYYY' ]})
```

- Beispiel:

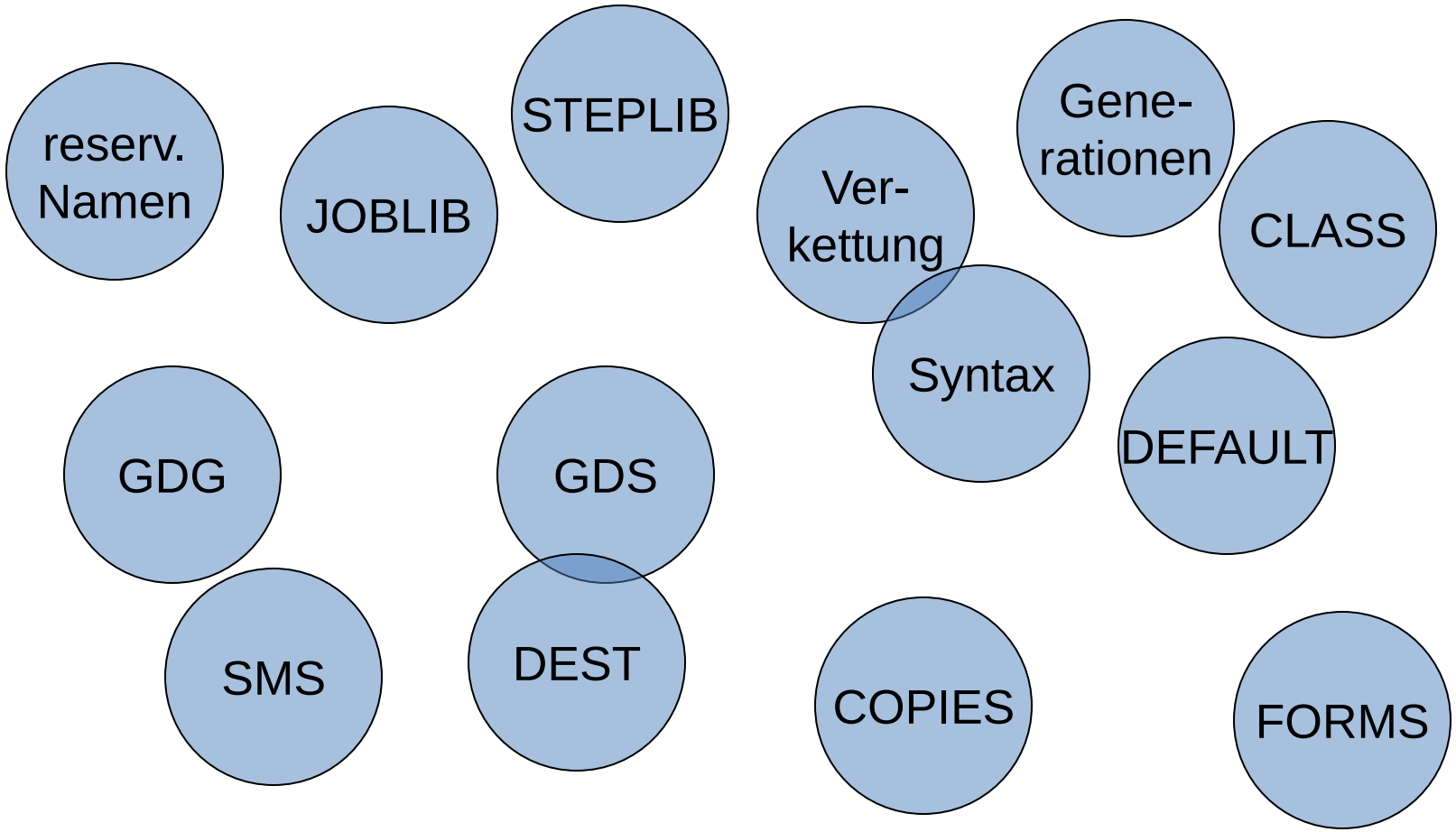
```
// SCHEDULE STARTBY=('14:50','12/13/2015'),
//       HOLDUNTL=('12:50','12/13/2015')
```

Inhalt

- Einführung
- Job-Beschreibung, Step-Beschreibung
- Datei-Beschreibung (1)
- Datei-Beschreibung (2)
- Standard- und Dienstprogramme - Überblick
- Job-Steuerung, Step-Steuerung
- Datei-Beschreibung (3)
- Include-Gruppe, JCL-Prozedur
- Diskussion und Austausch

Datei-Beschreibung (3)

Begriffe



Datei-Beschreibung (3)

reservierte DD-Namen – Überblick

- SYSMDUMP - Dump
- SYSABEND, SYSUDUMP - formatierter Dump
- SYSCHK, SYSCHKEOV - Checkpointing
- JOBCAT, STEPCAT - Katalogangabe
- JOBLIB, STEPLIB - Ladebibliotheken

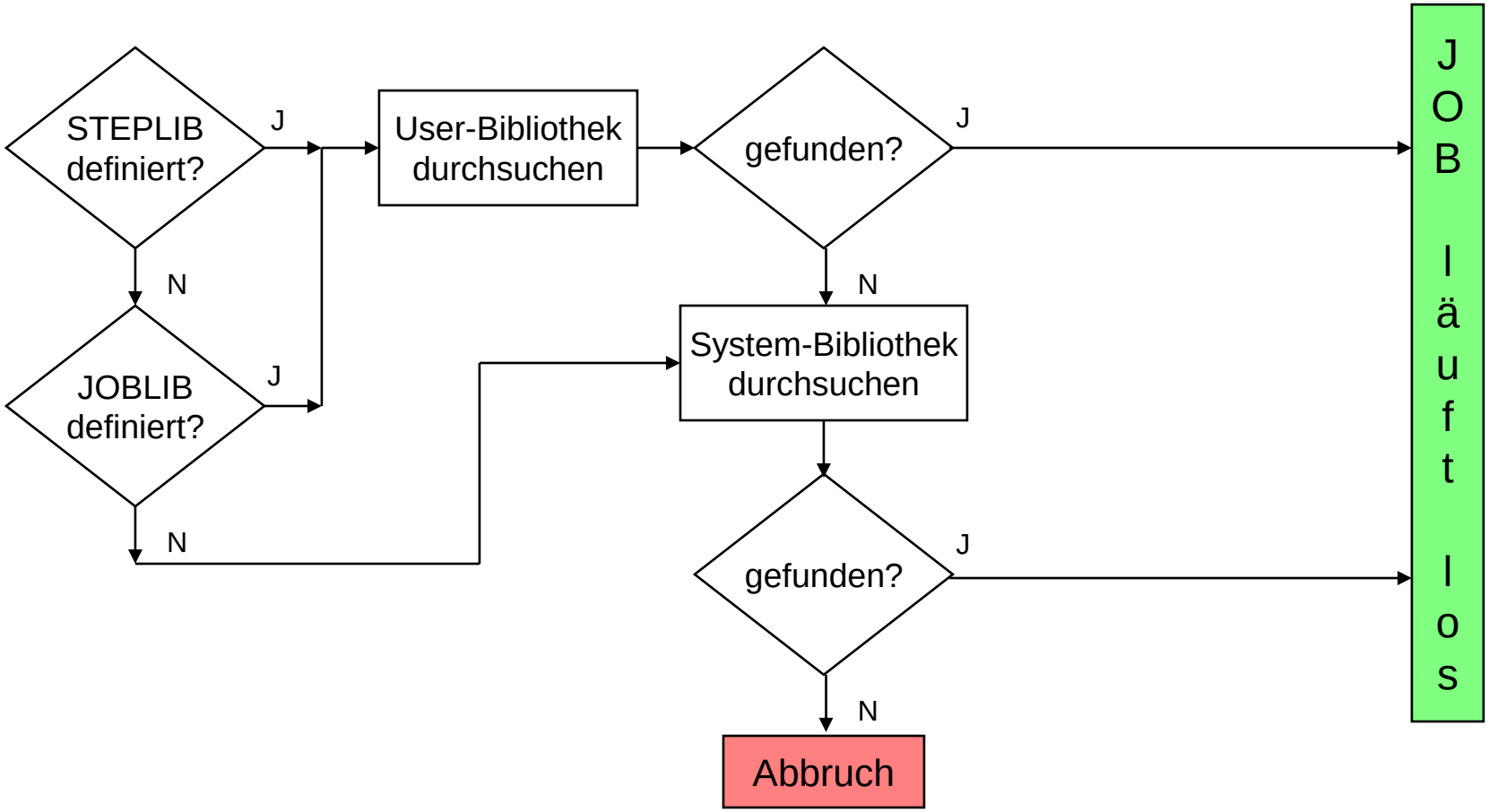
Datei-Beschreibung (3)

reservierte DD-Namen – JOBLIB, STEPLIB

- Syntax
  - //JOBLIB DD DISP=SHR,DSN=bibliothek1
  - //STEPLIB DD DISP=SHR,DSN=bibliothek2

Datei-Beschreibung (3)

reservierte DD-Namen – JOBLIB, STEPLIB (PAP)



Datei-Beschreibung (3)

cps4it

Übung(en)

- Kapitel 7.1: Anlegen PO-Datei als Ladebibliothek
- Kapitel 7.2: Job erstellen mit STEPLIB
- Kapitel 7.3: Job erstellen mit JOBLIB
- Kapitel 7.4: Job erstellen mit falschem Pgmname

- jeweils
  - Job wegschicken
  - Output analysieren

Datei-Beschreibung (3)

Verketten – Überblick

- Syntax
  - //ddname DD etc.
  - // DD etc.
  - // DD etc.
- Regeln
  - DSORG gleich (PS oder PDS evtl. PS+PDS-Mem)
  - RECFM gleich
  - LRECL gleich bei FB / Achtung bei VB
  - BLKSIZE der ersten Datei = max.

Datei-Beschreibung (3)

Verketten – Einsatz

- PS-Datei
  - für Anwendungsprogramm ist es logisch eine einzige Datei
  - EOF nach der letzten Datei
- PDS-Datei
  - für Anwendungsprogramm eine Bibliothek mit mehreren Ebenen
  - Anzahl Extents max. 123

Datei-Beschreibung (3)

Generationsdateien – Überblick

- Verwaltung von Dateien, die regelmäßig erstellt bzw. bearbeitet werden
- Generationsnummern
  - (0) aktuelle Generation
  - (-1) vorherige Generation bis (-n)
  - (+1) neu zu erstellende Generation; diese muss katalogisiert werden
- absolute Bezeichnungen möglich
  - DSN=datei.G3459V00

Datei-Beschreibung (3)

Generationsdateien – GDG

- zu GDS muss eine GDG existieren
- Anzahl Generationen: bis 255 (ab z/OS 2.2: 999)

- Beispiel:

```
//XV10733   JOB etc.
//*------------- Anlegen GDG
//DEFGDG    EXEC PGM=IDCAMS
//SYSPRINT DD SYSOUT=*
//SYSIN     DD *
  DEFINE GDG (NAME(XV10733.LAGER.BESTAND) -
              LIMIT(012) SCRATCH)
```

Datei-Beschreibung (3)

Generationsdateien – GDS

- Modellsatz
  - heute nicht mehr notwendig wegen SMS

- Beispiel:

```
//STEP1   EXEC PGM=P88N991
//DDSE01  DD DISP=SHR,DSN=XV10733.LAGER.BESTAND(0)
//DDSE02  DD DISP=SHR,DSN-XV10733.AEND
//DDSA01  DD DSN=XV10733.LAGER.BESTAND(+1),
//        DISP=(,CATLG,CATLG),UNIT=SYSDA,SPACE=(CYL,1),
//        DCB=XV10733.LAGER.MODELL   oder
//        DCB=RECFM= etc.
```

Datei-Beschreibung (3)

Generationsdateien – weitere Möglichkeiten

- Nutzen aktuelle bzw. vorige Generation
  - mit DSN=<name>(0),DISP=... bzw.
  - mit DSN=<name>(-1),DISP=...
- Nutzen alle Generationen
  - mit DSN=<name>,DISP=...
- Ändern GDG-Definitionen
  - mit IDCAMS und ALTER-Befehl
- Löschen GDG
  - mit IDCAMS und DELETE-Befehl

Datei-Beschreibung (3)

Übung(en)

- Kapitel 7.5: Anlegen GDG-Base-Entry
- Kapitel 7.6: Kopieren Instream-Daten -> GDS (3*)
- Kapitel 7.7: Ausdrucken GDS(aktuell)
- Kapitel 7.8: Ausdrucken alle GDSe
- Kapitel 7.9: Löschen GDG mit allen DSN

- jeweils
  - Job wegschicken
  - Output analysieren

Datei-Beschreibung (3)

OUTPUT-Anweisung – Überblick

- Syntax
  - //name OUTPUT parameter[,parameter]

- Parameter
  - DEFAULT=(YES|NO)
  - CLASS=(class|*)
  - FORMS=(formname)
  - COPIES=(nnn)
  - DEST=ziel

Datei-Beschreibung (3)

OUTPUT-Anweisung – Überblick – Beispiel

- Beispiel:

```
//OUT1      OUTPUT  COPIES=2
//OUT2      OUTPUT  DEST=EH2SEIR
//OUT3      OUTPUT  DEFAULT=YES,CLASS=*,COPIES=2
//OUT4      OUTPUT  CLASS=*,DEST=EH2SECX
//STEP1     EXEC    PGM=PROGX
//SYSPRINT  DD      SYSOUT=(,),OUTPUT=*.OUT2
//SYSUT2    DD      SYSOUT=T
//SYSUT3    DD      SYSOUT=(,),OUTPUT=*.OUT4
```

Datei-Beschreibung (3)

OUTPUT-Anweisung – CLASS, FORMS, COPIES, DEST

- CLASS
  - (Druck)Ausgabeklasse
- FORMS
  - Formularname
- COPIES
  - Anzahl der Kopien
- DEST
  - Druckername
- und . . . TITLE, NAME, ADDRESS, DEPT, BUILDING, ROOM, etc.

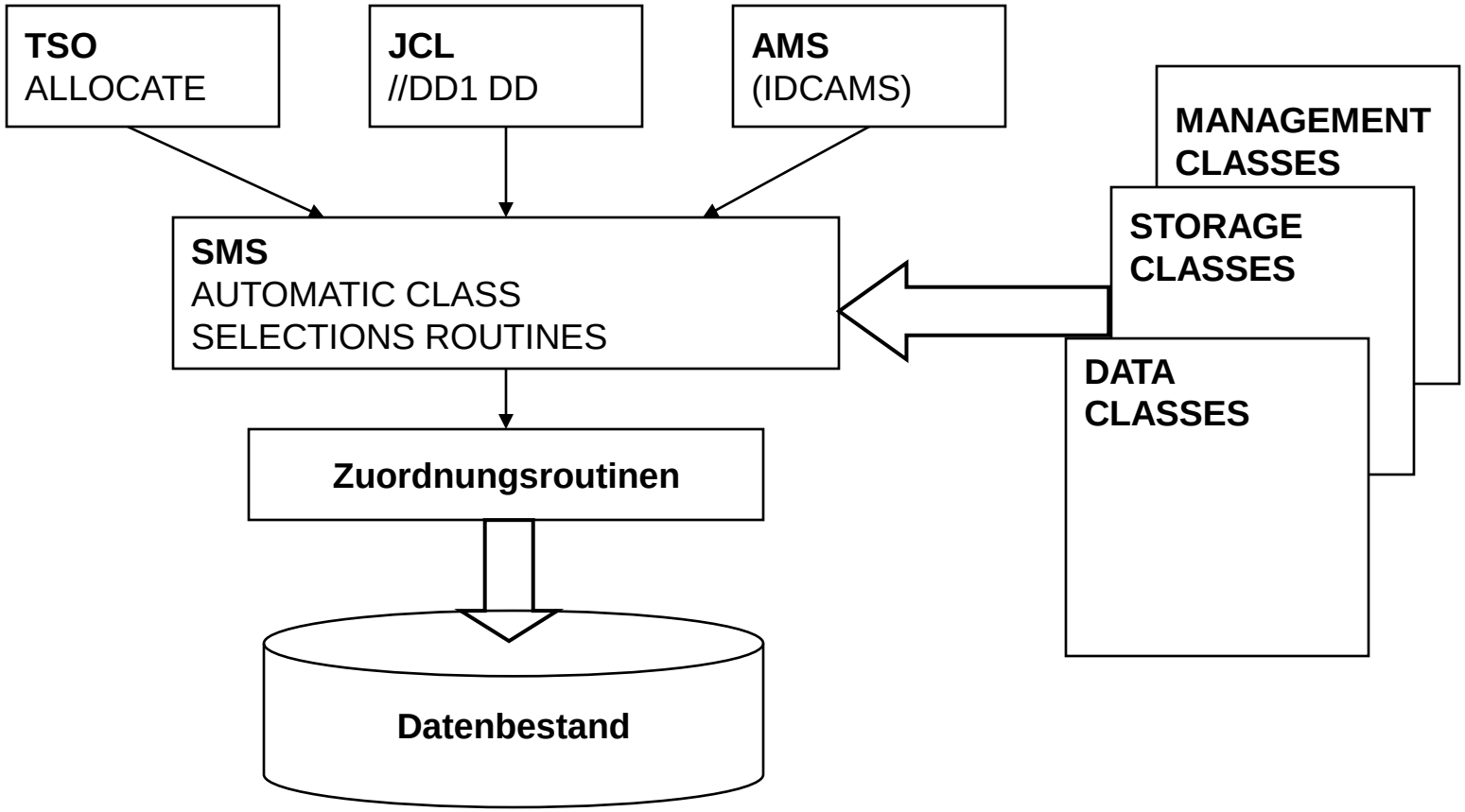
Datei-Beschreibung (3)

OUTPUT-Anweisung – only to confuse the russians . . .

```
//XV10733A JOB CLASS=<class>,MSGCLASS=<msgclass> . . .
//outname1 OUTPUT  DEFAULT=Y,CLASS=*,COPIES=<z1>
//outname2 OUTPUT  CLASS=<clsout2>,COPIES=<z2>
//outname3 OUTPUT  CLASS=<clsout3>,COPIES=<z3>
//stepnam1 EXEC    PGM=<pgm1>
//outnam11 OUTPUT  DEFAULT=Y,CLASS=<clsout11>
//outnam12 OUTPUT  CLASS=<clsout12>,COPIES=<z12>
//outnam13 OUTPUT  CLASS=<clsout13>,COPIES=<z13>
//ddname11 DD      SYSOUT=<cls11>
//ddname12 DD      SYSOUT=<cls12>,OUTPUT=*.<outnam12>
//ddname13 DD      SYSOUT=(,),OUTPUT=*.<outnam13>
//stepnam2 EXEC    PGM=<pgm2>
//outnam23 OUTPUT  CLASS=<clsout23>,COPIES=<z23>
//ddname21 DD      SYSOUT=<cls21>
//ddname22 DD      SYSOUT=<cls22>,OUTPUT=*.<stepnam1>.<outnam12>
//ddname23 DD      SYSOUT=(,),OUTPUT=(*.<outname2>,*.<outname3>)
```

Datei-Beschreibung (3)

SMS-Speicherverwaltung – Überblick



Datei-Beschreibung (3)

SMS-Speicherverwaltung – DATACLAS

DATA CLASS Attribute (Auswahl)								
Name	RECORG	RECFM	LRECL	KEYLEN	KEYOFF	AVGREC	VALUE	CISIZE
DATAFB	-	FB	80	-	-	U	80	-
DATAVB	-	VB	255	-	-	U	255	-
DATAPO	-	FB	80	-	-	U	80	-
LOADLIB	-	-	U	-	-	U	23476	-
DATAKS	KS	-	-	-	0	U	4096	4096
DATAES	ES	-	-	-	-	U	4096	4096

Datei-Beschreibung (3)

SMS-Speicherverwaltung – RECFM, RECORG, LRECL etc.

- RECFM: nicht VSAM
- RECORG: VSAM
- LRECL:
  - exakte Länge bei RECFM=F|FB
  - maximale Länge bei RECFM=V|VB
- KEYLEN, KEYOFF: nur VSAM

Datei-Beschreibung (3)

SMS-Speicherverwaltung – SPACE, AVGREC

- Verwendung
  - Festlegen des Platzbedarf
- Syntax
  - SPACE=(arlen,(prim[,sec])[,RLSE]],AVGREC=[U|K|M]
  - SPACE=(arlen,(prim[,sec],dir)[,RLSE]],AVGREC=[U|K|M]
- Definition
  - arlen durchschnittliche Satzlänge
  - prim anfängliche Anzahl Sätze
  - sec optionale Erweiterungsmenge
  - U,K,M Faktor=1,1K,1M

Datei-Beschreibung (3)

SMS-Speicherverwaltung – DSNTYPE

PO / PDS	PDSE / LIBRARY
Kann vom SMS verwaltet werden	Kann vom SMS verwaltet werden
16 Extents	123 Extents
Feste Directory-Größe (256 Byte pro Block)	Keine Begrenzung der Directory-Größe.
Etwa 5-20 Member-Einträge pro Dir-Block	Ingesamt über 500.000 Member-Einträge
Sequentiell organisierte Datei	Indizierte Directory
Wählbare Blockgröße Einheiten abhängig	Blockgröße wird Einheiten unabhängig vom SMS bestimmt
Compress notwendig	Dynamische Platzordnung; kein Compress notwendig
Beispiel:	

```
//AUSG     DD DSN=XV10733.JCL.CNTL,DISP=(,CATLG,DELETE),
//          DATACLAS=DATAPO,DSNTYPE=LIBRARY
```

Neu mit z/OS V2.x: DSNTYPE=(LIBRARY,2)
(Benutzung wird empfohlen / auch in ISPF3.2)

Beispiel siehe nächste Seite

Datei-Beschreibung (3)

SMS-Speicherverwaltung – LIKE, REFDD

- SYNTAX
  - LIKE=datasetname

Beispiele:
- JCL402
- JCL601N

- Beispiel:

```
//AUSG    DD DISP=(,CATLG,DELETE),
//           LIKE=XV10733.KUND.BASIS,
//           DSN=XV10733.KUND.BEST
```

Datei-Beschreibung (3)

SMS-Speicherverwaltung – STORCLAS

STORAGE CLASS - Attribute (Auswahl)						
Name	Dir Response µsec	Use (Read / Write)	Seq. Response µsec	Use (Read / Write)	Availibility	Guaranteed Space
STANDARD	-	-	10	-	STANDARD	NO
KRITISCH	10	-	10	-	STANDARD	NO
DBSTD	-	-	-	-	STANDARD	YES
DBFAST	10	R	10	R	CONTINOUS	YES
FASTREAD	5	R	5	R	STANDARD	NO
FASTWRIT	5	W	5	W	STANDARD	NO

- Beispiel:

```
//AUSG    DD DSN=XV10733.KUND.BEST,DISP=(,CATLG,DELETE),
//           DATACLAS=DATAFS,STORCLAS=STANDARD
```

Datei-Beschreibung (3)

SMS-Speicherverwaltung – MGMTCLAS

MANAGEMENT CLASS - Attribute (Auswahl)						
Name	Expire Non-Usage	Expire Date / Days	Max. Rentention Period	Partial Release (Unused Space)	Primary Days	Auto Backup
STANDARD	NOLIMIT	NOLIMIT	NOLIMIT	YES	15	Y
NOACTION	NOLIMIT	NOLIMIT	NOLIMIT	NO	2	Y
GDG	NOLIMIT	NOLIMIT	NOLIMIT	YES	15	Y
DBSTD	NOLIMIT	NOLIMIT	NOLIMIT	NO	2	Y
DBMIGRAT	NOLIMIT	NOLIMIT	NOLIMIT	NO	15	Y

- Beispiel:

```
//AUSG    DD DSN=XV10733.KUND.BEST,DISP=(,CATLG,DELETE),
//           DATACLAS=DATAFS,STORCLAS=STANDARD,
//           MGMTCLAS=STANDARD
```

Datei-Beschreibung (3)

cps4it

Übung(en)

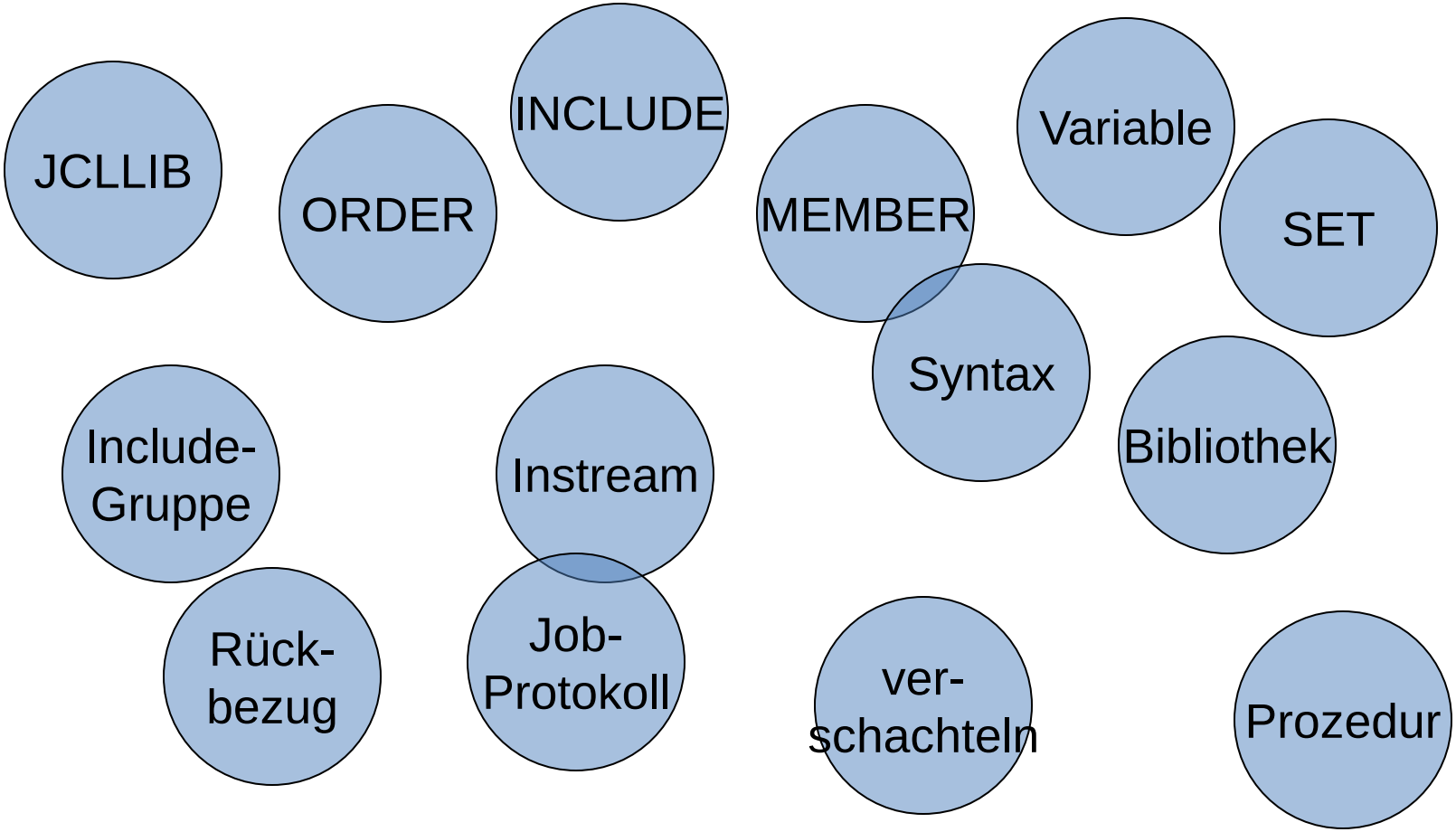
- Kapitel 7.10: welche STORCLAS in Firma
- Kapitel 7.11: welche MGMTCLAS in Firma

Inhalt

- Einführung
- Job-Beschreibung, Step-Beschreibung
- Datei-Beschreibung (1)
- Datei-Beschreibung (2)
- Standard- und Dienstprogramme - Überblick
- Job-Steuerung, Step-Steuerung
- Datei-Beschreibung (3)
- Include-Gruppe, JCL-Prozedur
- Diskussion und Austausch

Include-Gruppe, JCL-Prozedur

Begriffe



Include-Gruppe, JCL-Prozedur

Allgemeines – Überblick

- Definition
  - Sequenz von JCL-Anweisungen
  - wieder verwendbar
  - zentrale Speicherung möglich
- Parameter und/oder DD-Anweisungen
  - ergänzen
  - überschreiben
  - aufheben

Include-Gruppe, JCL-Prozedur

JCLLIB-Anweisung – Überblick

- Verwendung
  - stehen allen Jobs zur Verfügung
  - im Normalfall kein Schreibrecht
  - eigene Bibliotheken möglich
  - nach JOB-Anweisung
  - vor 1. EXEC-Anweisung

Include-Gruppe, JCL-Prozedur

JCLLIB-Anweisung – ORDER

- Syntax
  - //[name] JCLLIB ORDER=bibliothek
    oder
  - //[name] JCLLIB ORDER=(bibl1[,bibl2,bibl3]...)

- Beispiel:

```
//JOBX     JOB  ,SEIDLER,. . .
//PROZLIB  JCLLIB ORDER=XV10733.PROZ.JCL
//STEP1    EXEC PROC=PROZ1
```

Include-Gruppe, JCL-Prozedur

INCLUDE-Anweisung – Überblick

- Verwendung
  - fast alle JCL-Anweisungen möglich außer:
    - JOB
    - PROC / PEND
    - JCLLIB
    - DD * bzw. DD DATA
  - bis 15 INCLUDE-Anweisungen verschachteln

Include-Gruppe, JCL-Prozedur

INCLUDE-Anweisung – Member – 1

- Beispiel:
  - JCLGRP1 in XV10733.PROZ.JCL

```
//STEP1    EXEC PGM=IEBGENER
//SYSUT1   DD DISP=SHR,DSN=XV10733.TEST1.DATA
//SYSUT2   DD DISP=SHR,DSN=XV10733.TEST2.DATA
//SYSPRINT DD SYSOUT=*
//SYSIN    DD DUMMY
```

  - Übernahme der INCLUDE-Gruppe

```
//JOBX     JOB  ,SEIDLER,. . .
//BIBL1    JCLLIB ORDER=XV10733.PROZ.JCL
//         INCLUDE MEMBER=JCLGRP1
```

Include-Gruppe, JCL-Prozedur

INCLUDE-Anweisung – Member – 2

- Beispiel – Ergebnis:
  - JCLGRP1 in XV10733.PROZ.JCL

```
//JOBX      JOB  ,SEIDLER,. . .
//BIBL1     JCLLIB ORDER=XV10733.PROZ.JCL
//           INCLUDE MEMBER=JCLGRP1
XXSTEP1     EXEC PGM=IEBGENER
XXSYSUT1    DD DISP=SHR,DSN=XV10733.TEST1.DATA
XXSYSUT2    DD DISP=SHR,DSN=XV10733.TEST2.DATA
XXSYSPRINT DD SYSOUT=*
XXSYSIN     DD DUMMY
```

Include-Gruppe, JCL-Prozedur

Variable, SET – Überblick

- Parametrisierung der Prozedur
- maximal 8 Stellen
- & vorangesetzt bei Benutzung
- kein & vorangesetzt bei Belegung
- Festlegung des Wertes
  - mit SET-Anweisung vor erster Verwendung
  - als Defaultwert für Prozeduren
  - als Parameter bei Aufruf der Prozedur

Include-Gruppe, JCL-Prozedur

Variable, SET – Verwendung von Variablen

- Variablen sind Zeichenketten, max 255 Byte
- nur in Parameterfeld von JCL-Anweisungen
- können einzeln stehen
- können kombiniert werden (Achtung .)
- Systemvariablen sind immer verfügbar

Include-Gruppe, JCL-Prozedur

Variable, SET – SET-Anweisung – 1

- Beispiel:
  - JCLGRP2 in XV10733.PROZ.JCL

```
//STEP1    EXEC PGM=IEBGENER
//SYSUT1   DD DISP=&STATUS,DSN=&NAME
//SYSUT2   DD DISP=SHR,DSN=XV10733.TEST2.DATA
//SYSPRINT DD SYSOUT=*
//SYSIN    DD DUMMY
```

  - Übernahme der INCLUDE-Gruppe

```
//JOBX     JOB  ,SEIDLER,. . .
//BIBL1    JCLLIB ORDER=XV10733.PROZ.JCL
//         SET    NAME=XV10733.TEST1.DATA,STATUS=OLD
//         INCLUDE MEMBER=JCLGRP2
```

Include-Gruppe, JCL-Prozedur

Variable, SET – SET-Anweisung – 2

- Beispiel – Ergebnis:

```
//JOBX      JOB  ,SEIDLER,. . .
//BIBL1     JCLLIB ORDER=XV10733.PROZ.JCL
//          INCLUDE MEMBER=JCLGRP2
XXSTEP1     EXEC PGM=IEBGENER
XXSYSUT1    DD DISP=OLD,DSN=XV10733.TEST1.DATA
XXSYSUT2    DD DISP=SHR,DSN=XV10733.TEST2.DATA
XXSYSPRINT DD SYSOUT=*
XXSYSIN     DD DUMMY
```

- SET wird immer wirksam, selbst innerhalb eines IF/THEN, das nicht durchlaufen wird.

Include-Gruppe, JCL-Prozedur

Nutzung Variable in instream

- Seit Kurzem ist die Nutzung in instream möglich

```
//VORNE    EXPORT SYMLIST=ALL
. . .
//         SET ZFSDSN=OMVS.INSTALL.TABEX.ZFS
//         SET ZFSMNTP='/install/nonsmpe/tabex/'
//         SET ZFSSPACE='1500,0'
//DEFINE EXEC PGM=IDCAMS
//SYSPRINT DD SYSOUT=*
//SYSIN DD *,SYMBOLS=JCLONLY
  DEFINE CLUSTER (                    +
           NAME(&ZFSDSN.)             +
           LINEAR CYL(&ZFSSPACE.)     +
           DATACLASS(EXTENDAD)        +
           SHAREOPTIONS(3) )
```

Include-Gruppe, JCL-Prozedur

Nutzung Variable in instream - Hinweise

- EXPORT: Man kann auch einzelne Variablen explizit auswählen
- Die Nutzung der Variablen ist auch in Prozeduren, Includes … möglich. Werte können in der Hierarchie festgelegt warden und von da an genutzt werden.
- JCK kennt den EXPORT Befehl zur Zeit noch nicht. (-> JCK HZT)

Include-Gruppe, JCL-Prozedur

JCL-Prozedur – Überblick

- Verwendung
  - ein oder mehrere vollständige Steps
  - fast alle JCL-Anweisungen möglich außer:
    - JOB
    - JCLLIB, JOBCAT, JOBLIB
    - ~~DD * bzw. DD DATA~~
- steht in
  - Standardbibl (z.B. SYS1.PROCLIB)
  - Userbibl (mit JCLLIB definiert)
  - Job (Instream-Prozedur)

Include-Gruppe, JCL-Prozedur

JCL-Prozedur – Definition von Bibliotheksprozeduren

- Syntax
  - [//[procname] PROC [default-werte]]
  - //procstepnm EXEC PGM=pgmname
  - // <jcl-anweisungen>
  - [//[name] PEND]
- Name der Prozedur = Membername in Bibl. !!

Include-Gruppe, JCL-Prozedur

JCL-Prozedur – Definition der Instream-Prozedur

- Syntax
  - //jobname JOB ,name, . . .
  - //procname PROC [default-werte]
  - //procstepnm EXEC PGM=pgmname
  - // <jcl-anweisungen>
  - //[name] PEND
  - . . .
  - //STEP1 EXEC procname,[var-zuweisungen]
- Beschreibung der Proc *vor* dem Aufruf.

Include-Gruppe, JCL-Prozedur

JCL-Prozedur – Benutzung der Prozedur

- beliebig oft aufrufbar in einem Job
- bis zu 15 Aufrufe verschachteln
- Syntax:

```
//stepname EXEC [PROC=]procname
//procstep.ddname1  DD     <modifiz. DD-Anweisung>
//procstep.ddname2  DD     <zusätzl. DD-Anweisung>
//procstep.STEPLIB  DD
//                  DD
//                  DD     <modifiz. DD-Anweisung>
//                  DD
```

Include-Gruppe, JCL-Prozedur

JCL-Prozedur – Aufruf-Parameter – 1

- Beispiel:
  Prozedur PROC01 in T98MVS.JCLLIB

```
//PROCXX   PROC KLASSE=Z,EINDAT=NULLFILE,
//              CPYDAT=NULLFILE,ANZ=
//COPY     EXEC PGM=IEBGENER
//SYSIN    DD   DUMMY
//SYSPRINT DD   SYSOUT=&KLASSE,COPIES=&ANZ
//SYSUT1   DD   DISP=SHR,DSN=&EINDAT
//SYSUT2   DD   DISP=OLD,DSN=&CPYDAT
//SORT     EXEC PGM=ICEMAN,TIME=(1,0),COND=(0,NE,COPY)
//SYSIN    DD   DISP=SHR,DSN=K.L.M(SORT01)
//SYSOUT   DD   SYSOUT=&KLASSE,COPIES=&ANZ
//SORTIN   DD   DISP=SHR,DSN=&CPYDAT
//SORTOUT  DD   DISP=OLD,DSN=&AUSDAT
//             PEND
```

Include-Gruppe, JCL-Prozedur

JCL-Prozedur – Aufruf-Parameter – 2

- Aufruf der Prozedur PROC01 in T98MVS.JCLLIB

```
//JOBDEMO1 JOB  CLASS=A,MSGCLASS=*
//         JCLLIB ORDER=(T98MVS.JCLLIB)
[//          SET ANZ=3]
//STEP01   EXEC PROC=PROC01,KLASSE='*',
//              EINDAT='T98MVS.UMSATZ.FDKOELN',
//              CPYDAT='T98MVS.UMSATZ.KOPIE',
//              AUSDAT='T98MVS.UMSATZ.FDKOELN.SORT',
//              TIME.COPY=(3,0),TIME.SORT=,
//              COND.SORT=(4,LT,COPY)
```

- Regeln für Parameter beachten

Include-Gruppe, JCL-Prozedur

JCL-Prozedur – Aufruf-Parameter – 3

- Arten und Wirkung
  - Wertzuweisung an symbolische Variablen
  - Wertzuweisung an Schlüsselwortparm in EXEC-Anw
    - für jeden Prozedurstep oder
    - für einen bestimmten Prozedurstep
  - Wertzuweisungen bewirken
    - Ergänzung: von bisher unbestimmten Variablen
    - Ersetzung: von schon besetzten Variablen
    - Aufhebung: von besetzten Variablen

Include-Gruppe, JCL-Prozedur

JCL-Prozedur – Aufruf-Parameter – 4

- Regeln für Variablen
  - Zeichenketten mit Sonderzeichen brauchen ‘ ‘
  - Wertzuweisungen > Defaultwert > SET (> Fehler)
- Regeln für Schlüsselwortparameter
  - Wertzuweisung in der Reihenfolge
    - Werte, die für jeden Step in Prozedur gelten
    - Werte, die für eine Prozedur gelten, in der Reihenfolge der Steps in der Prozedur

Include-Gruppe, JCL-Prozedur

JCL-Prozedur – Aufruf-Parameter – 5

- Sonderregeln
  - PGM= nicht veränderbar
  - TIME= ohne Prozedurstep gilt für alle Steps
  - PARM= ohne Prozedurstep gilt für 1. Step

Include-Gruppe, JCL-Prozedur

JCL-Prozedur – Aufruf mit OUTPUT und DD

- Wirkung
  - Ergänzen, Ersetzen, Aufheben, Widerspruch (nur DD-Anweisungen)
- Regeln
  - procstepname.ddname bzw.
  - procstepname.outname
  - zuerst Modifikationen
    - Reihenfolge in Prozedur einhalten
  - danach Hinzufügungen
    - Reihenfolge in Prozedur

Vermeide Anweisungen ohne Prozedurstep !!

Include-Gruppe, JCL-Prozedur

JCL-Prozedur – spezielle Möglichkeiten bei DD

- ~~DD * bzw. DD DATA~~
  - ~~weglassen in Prozedur / bei Aufruf hinzufügen~~
  - ~~formulieren mit DD DUMMY / bei Aufruf modifizieren (mit Widerspruch)~~
  - ~~DD-Namen als Datei definieren und aufrufen~~
- Verkettung
  - komplette Verkettung in Prozedur bei Aufruf
  - aber unveränderte Parameter leer lassen
  - zusätzliche Angaben hinzufügen
  - !! DD DUMMY beendet Verkettung

Include-Gruppe, JCL-Prozedur

JCL-Prozedur – Rückbezug

- COND

```
//  COND=(0,NE,STEP01)
//  COND=(0,NE,STEP01.COPY)
```

- DD-Anweisungen

```
//  DSN=*.STEP01.SYSUT1
//  DSN=*.STEP01.COPY.SYSUT1
//  VOL=REF=*.STEP01.SYSUT1
//  VOL=REF=*.STEP01.COPY.SYSUT1
//  REFDD=*.STEP01.SYSUT1
//  OUTPUT=*.STEP01.COPY.SYSUT1
```

Include-Gruppe, JCL-Prozedur

JCL-Prozedur – verschachteln

- bis zu 15 Aufrufe von Prozeduren schachteln
- Variablenbesetzung
  - Proc-Aufruf > Default in Proc > SET-Anweisung > Wert der Variablen auf Aufruf-JCL > Fehler
- Modifikationen nur für nächste Ebene möglich
- kein Verschachteln von Proc-Definitionen

Include-Gruppe, JCL-Prozedur

JCL-Prozedur – Jobablauf-Protokoll

- Abhängigkeit von MSGLEVEL
  - MSGLEVEL=(1,x) heißt Ausgabe der JCLs
- Spalte 1-3
  - `//` JCL-Anweisung im rufenden Job
  - `xx` unverändert aus kat. Proc übernommen
  - `x/` modifiziert aus kat. Prozedur
  - `xx*` Kommentar in kat. Prozedur
  - `++` unveränderte Anweisung aus Instr.-Proc
  - `+/` modifiziert aus Instr.-Proc
  - `++*` Kommentar in Instr.-Proc
  - `***` Kommentar / Steueranweisung für JES

Include-Gruppe, JCL-Prozedur

JCL-Prozedur – Standard-Prozeduren

- siehe Firmenbibliothek

Include-Gruppe, JCL-Prozedur

Übung(en)

- Kapitel 8.1: Instream-Prozedur
- Kapitel 8.2: Instream-Prozedur - parametrisiert
- Kapitel 8.3: externe Prozedur
- Kapitel 8.4: externe Prozedur - parametrisiert
- Kapitel 8.5: Ü 8.1 bis 8.4 mit
  - unterschiedlichen MSGLEVEL
  - verschiedenen Modifikationen
  - verschiedenen Ergänzungen

Verschiedenes

Check leeres File (es gibt oft dafür spezielle Programme in Firmen)

```
//*-------------------------------------------------------------------*/
//* This JCL is to find if a file is empty or not. If there are no    */
//* records, the return-code will be 4.                               */
//*-------------------------------------------------------------------*/
//STEP01   EXEC PGM=IDCAMS
//SYSPRINT  DD SYSOUT=*
//IN1       DD DISP=SHR,DSN=your.dataset.name
//SYSIN     DD *
  PRINT INFILE(IN1) COUNT(001)
=====================================================================
//*-------------------------------------------------------------------*/
//* This JCL is to find if a file is empty or not using ICETOOL.      */
//* If there are no records, the return-code will be 12               */
//*-------------------------------------------------------------------*/
//STEP01  EXEC PGM=ICETOOL
//TOOLMSG  DD SYSOUT=*
//DFSMSG   DD SYSOUT=*
//IN       DD DISP=SHR,DSN=your.dataset.name
//TOOLIN   DD *
COUNT FROM(IN) EMPTY
```

Inhalt

- Einführung
- Job-Beschreibung, Step-Beschreibung
- Datei-Beschreibung (1)
- Datei-Beschreibung (2)
- Standard- und Dienstprogramme - Überblick
- Job-Steuerung, Step-Steuerung
- Datei-Beschreibung (3)
- Include-Gruppe, JCL-Prozedur
- Diskussion und Austausch