

z/OS Job Control Language

Grundlagen

cps4it


consulting, projektmanagement und seminare für die informationstechnologie

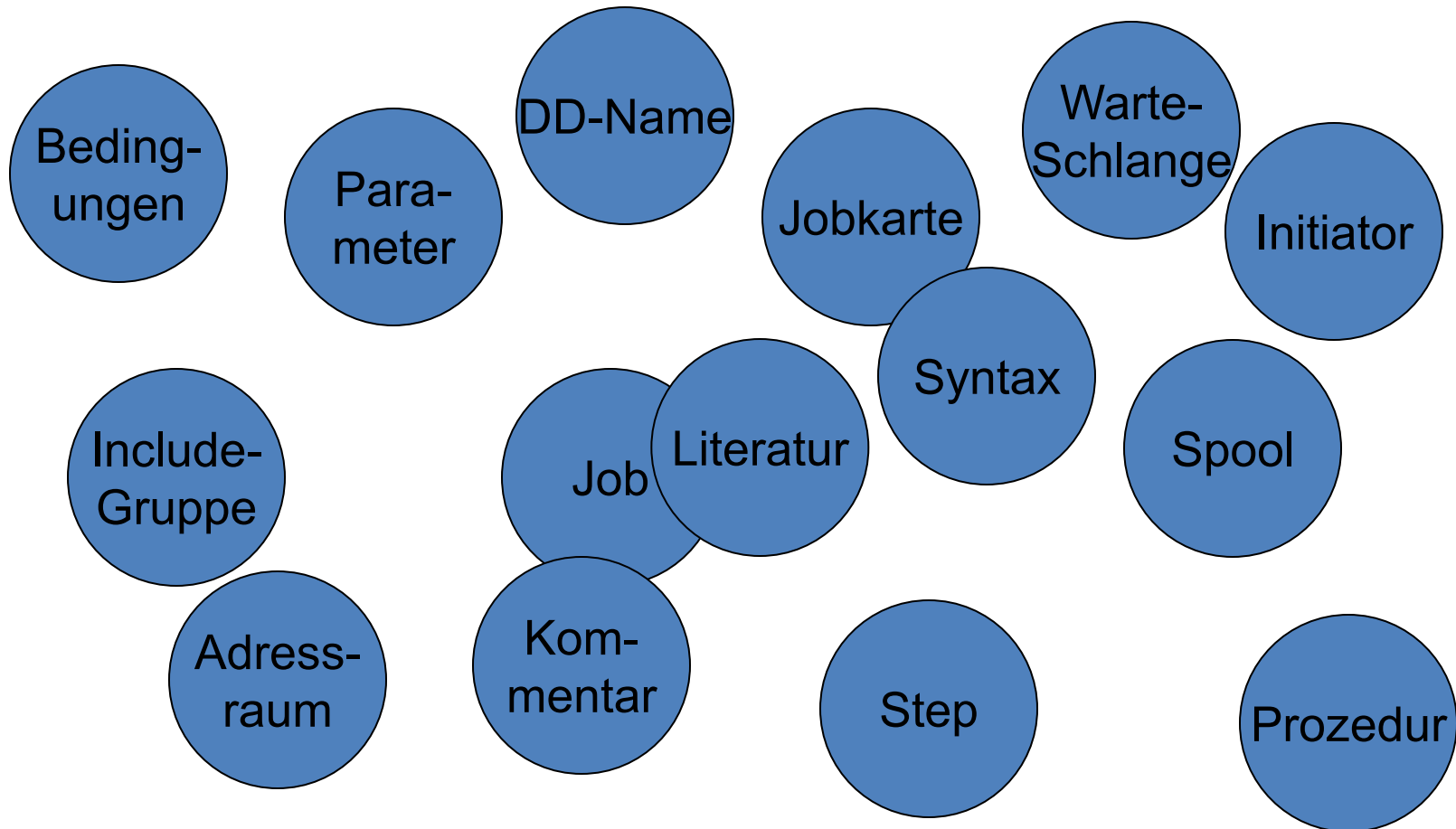
Ralf Seidler, Stromberger Straße 36A, 55411 Bingen

Fon: +49-6721-992611, Fax: +49-6721-992613, Mail: ralf.seidler@cps4it.de

Internet: <http://www.cps4it.de>

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

-
- 
- A blue arrow pointing to the right, highlighting the first item in the list.
- 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



- Bookmanager
 - local im Intranet oder im Internet bei IBM
- Internetseiten
 - siehe Suchmaschinen
 - (z.B. <http://www.isc.ucsb.edu/tsg/jcl.html>)
- G.D.Brown:
 - JCL – Jobcontrol Language im Betriebssystem z/OS
 - 4. Auflage
 - ISBN 978-3486273977



64,80€

JCL heißt ...

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

Job Control Language

```
//A12345XX JOB CLASS=A,MSGCLASS=T
//STEP01   EXEC PGM=KOPIEREN
//EINGABE  DD   DISP=SHR,DSN=X.Y.Z
//AUSGABE  DD   DISP=OLD,DSN=A.B.C
//STEP02   EXEC PGM=SORTIERE
//SYSIN    DD   DISP=SHR,DSN=A.B.C
//SYSOUT   DD   DISP=OLD,DSN=D.E.F
//STEP03   EXEC PGM=DRUCKEN
//INPUT    DD   DISP=SHR,DSN=D.E.F
//DRUCK    DD   SYSOUT=F
//SYSPRINT DD   SYSOUT=*
```

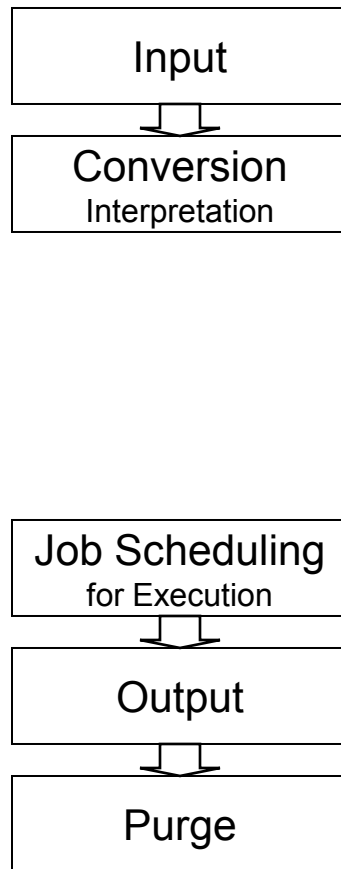
Diagram illustrating the structure of a Job Control Language (JCL) job. The job is divided into three steps, each indicated by a bracket on the right labeled "Step". The first step includes the job definition (JOB) and the first EXEC and DD statements. The second step includes the second EXEC and DD statements. The third step includes the third EXEC and DD statements. A large bracket on the far right groups all three steps under the label "Job".

Labels at the bottom of the diagram:

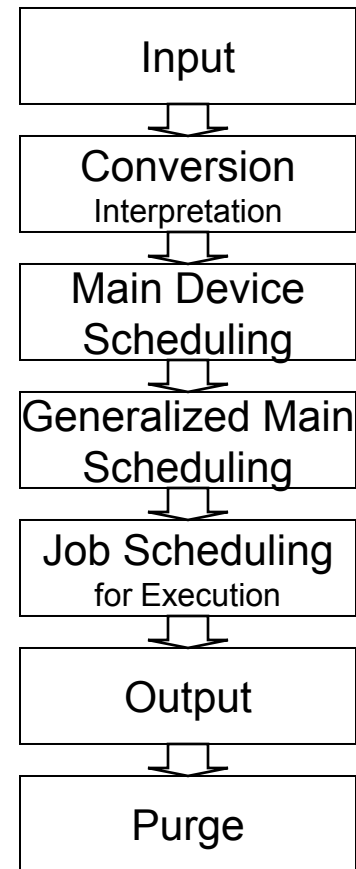
- Jobanweisung (points to the JOB statement)
- DD-Anweisung (points to the DD statements)
- Stepanweisung (points to the EXEC statements)

JES2 / JES3

JES2



JES3



Eingabe-, Ausgabe-Warteschlangen (1)

Warteschlangen

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

Adressräume

MAS- TER	SYS- TEM	INIT1 C=AC	INIT2 C=A	INIT3 C=C	INIT4 C=9ZE	INIT5 C=D	TSO

Eingabe-, Ausgabe-Warteschlangen (2)

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			

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

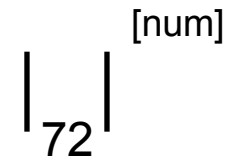
```
//name op-code [param1][,param2]... [kommentar]
```

| 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	

JCL-Anweisungen – Syntax – 2

//[name] op-code [param1][,param2]...



//[name] op-code [param1][,param2-1]...

* [num]

// [param2-2][,param3]...

[num]



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

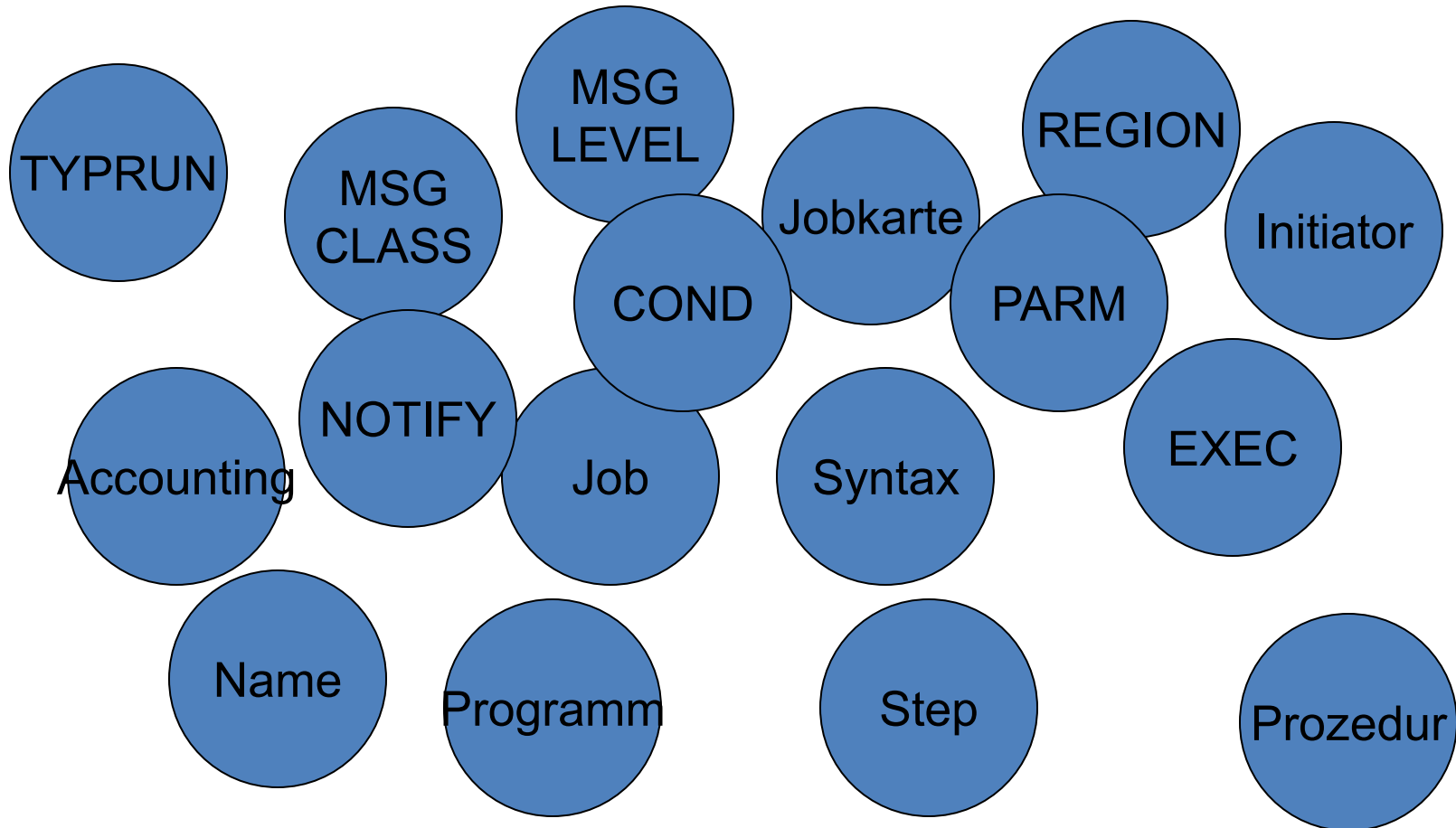
- Kapitel 1.1: Auswahl und Test User-ID
- Kapitel 1.2: Bibliothek erstellen



-
- 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-Anweisung

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

- Standards beachten

- 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'
```

- Syntax

- [,programmierer-name]
optional

- Beispiel:

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

```
//JOB2 JOB , ' SEIDLER'
```

```
//JOB3 JOB (CPS4IT,TRAINING) , 'MAYER&&HUBER'
```



Übung(en)

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



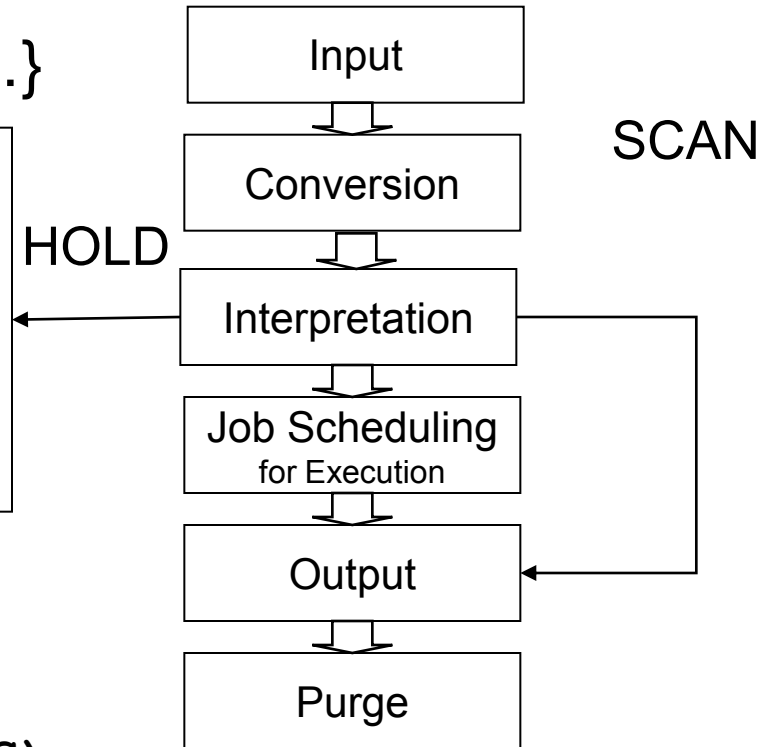
Job-Beschreibung, Step-Beschreibung

JOB-Anweisung – TYPRUN

- Syntax

– TYPRUN={HOLD|SCAN|...}
optional

<u>QUEUE</u>	<u>CLASS</u>
JOB1	C
JOB2	E
JOB3	C
JOB4	H
JOB5	E



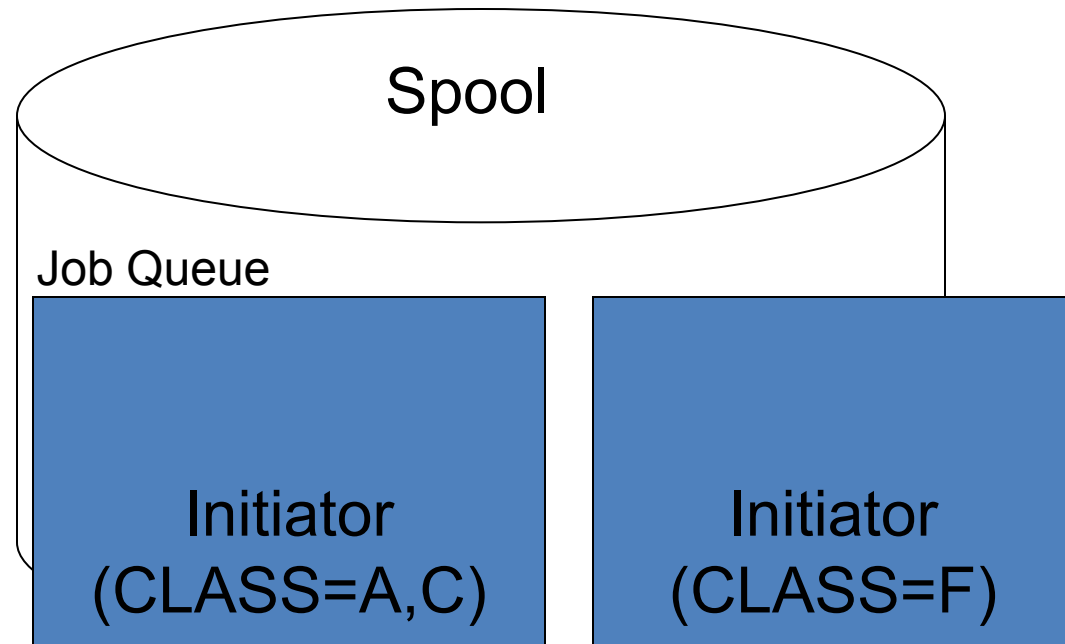
- Beispiel:

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

Job-Beschreibung, Step-Beschreibung

JOB-Anweisung – CLASS

- Syntax
 - CLASS=jobclass
optional



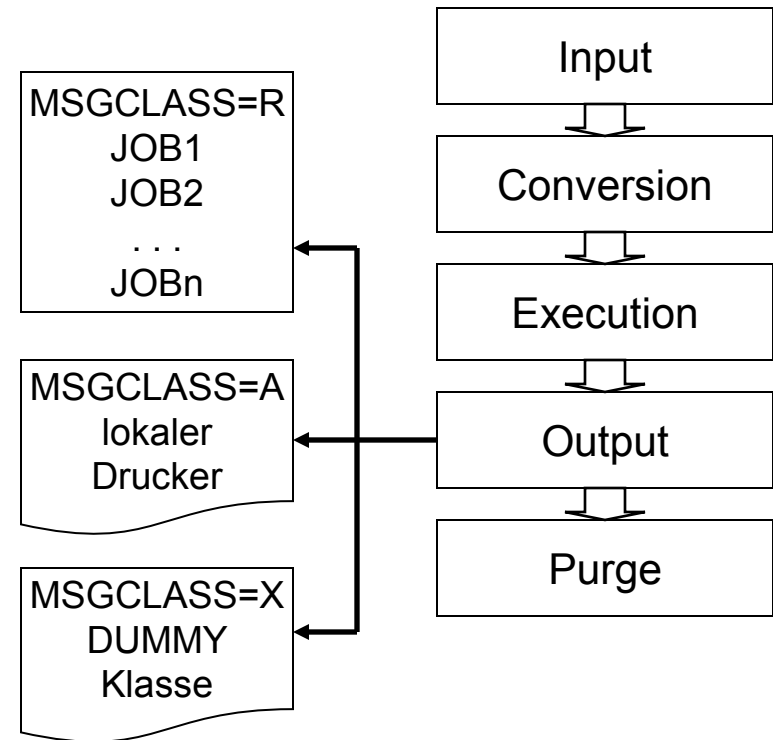
- Beispiel:

```
//JOB1 JOB (CPS4IT, TRAINING) , CLASS=A
```

```
//JOB2 JOB (CPS4IT, TRAINING) , CLASS=C
```

JOB-Anweisung – MSGCLASS

- Syntax
 - MSGCLASS=class
optional



- Beispiel:

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


- Syntax

- MSGLEVEL=([statements][,messages])

- statements $\in \{0,1,2\}$

- messages $\in \{0,1\}$

- (Die unterschiedlichen Auswirkungen sehen wir uns während der Übungen an.)

- Beispiel:

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



## Übung(en)

---

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



## JOB-Anweisung – TIME

---

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

- 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
```

- Syntax

– REGION=  $\left\{ \begin{array}{l} \text{nnnnnnnK} \\ \text{mmmmM} \end{array} \right\}$

- Beispiel:

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

## JOB-Anweisung – COND

---

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

- Beispiel:

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

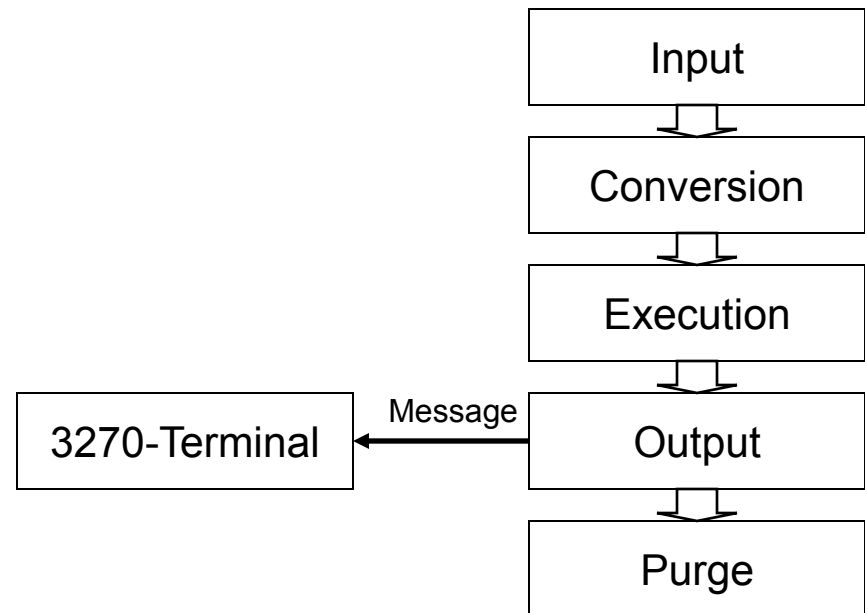
```
//JOBY JOB ,SEIDLER,CLASS=C,COND=(8,LT)
```

## JOB-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
```

- 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
```



## Übung(en)

---

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





## EXEC-Anweisung – Überblick – 1

---

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

## 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
```

```
. . .
```

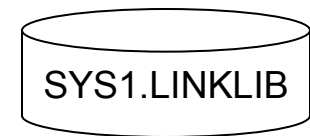
## EXEC-Anweisung – PGM

---

- 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
```



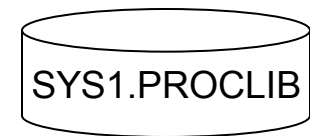
## EXEC-Anweisung – PROC

---

- Syntax
  - [PROC=]prozedur-name

- Beispiel:

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



## Übung(en)

---

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



## EXEC-Anweisung – PARM

---

- Syntax
  - PARM=information

- Beispiel:

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

## 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
```

## EXEC-Anweisung – REGION

---

- Syntax

– REGION= { nnnnnnK  
          mmmmM }

- Beispiel:

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



- Syntax

– COND= {  
    (code,op[,stepname])  
    EVEN  
    ONLY  
    ((code,op[,stepname]),(code,op[,stepname]))...[,EVEN]  
    ((code,op[,stepname]),(code,op[,stepname]))...[,ONLY] } }

- Detaillierung später



## Übung(en)

---

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

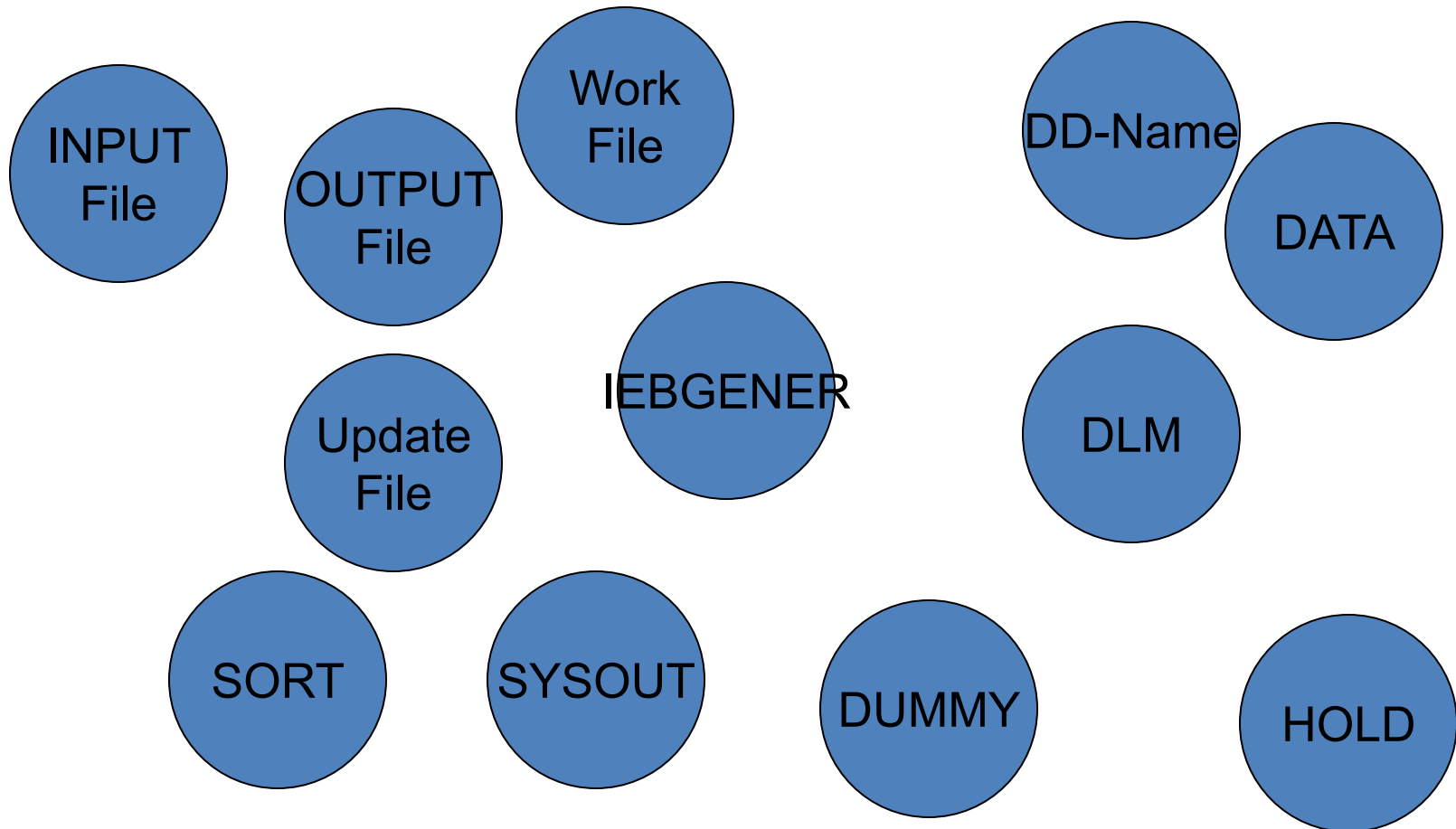


- 
- Einführung
  - Job-Beschreibung, Step-Beschreibung
  - • Datei-Beschreibung (1)
  - Datei-Beschreibung (2)
  - Standard- und Dienstprogramme - Überblick
  - Job-Steuerung, Step-Steuerung
  - Datei-Beschreibung (3)
  - Include-Gruppe, JCL-Prozedur
  - Diskussion und Austausch

# Datei-Beschreibung (1)

## Begriffe

---



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

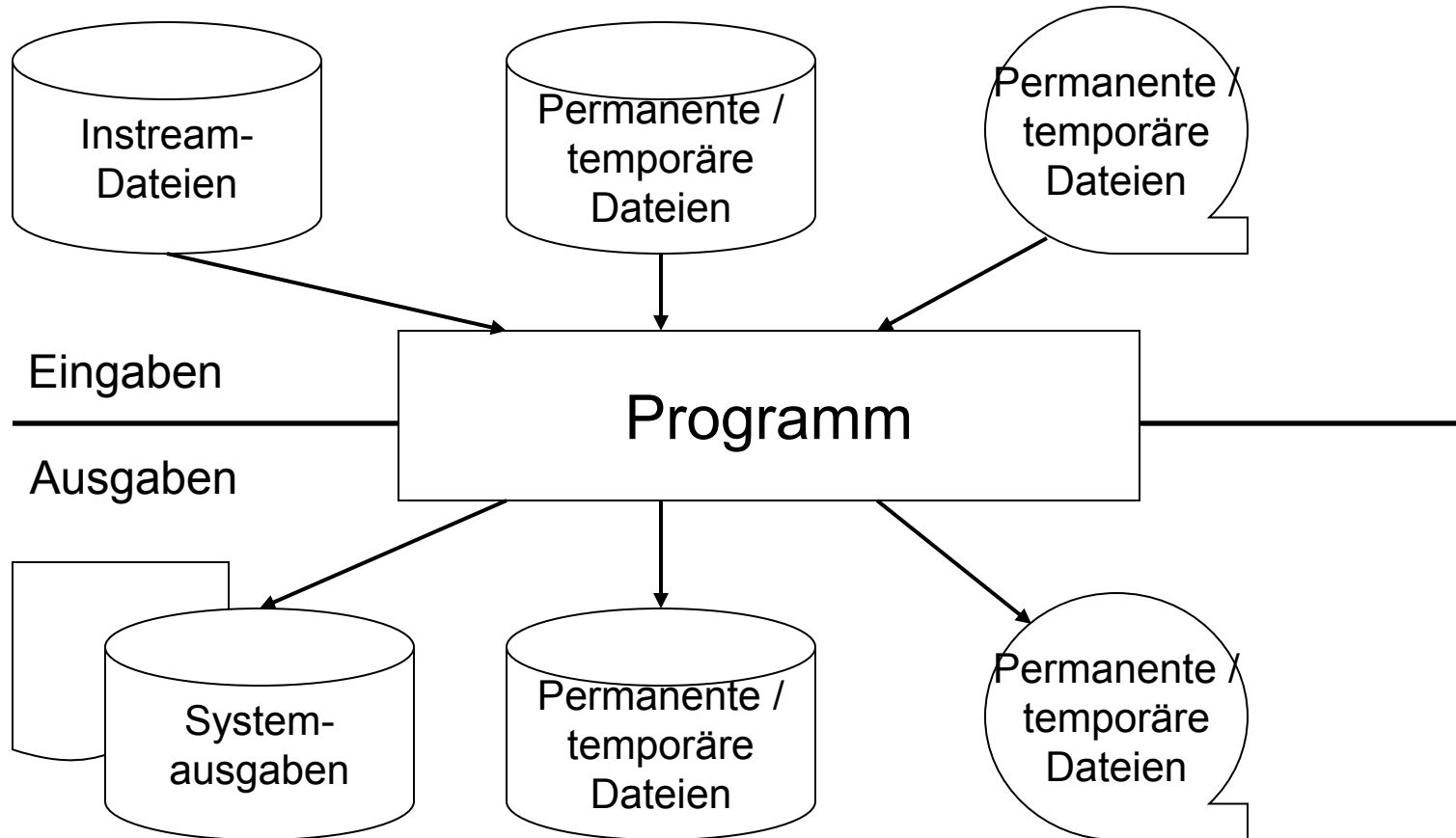
- COBOL
  - SELECT EINGABE ASSIGN TO LESE1
  - SELECT AUSGABE ASSIGN TO DRUCK1

- Beispiel:

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

# Datei-Beschreibung (1)

## DD-Anweisung (1) – Überblick



## DD-Anweisung – Felder

---

- Syntax

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

- Beispiel:

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



## System-Eingabe – DATA

---

- Syntax

–  
– //ddname DD  $\left\{ \begin{array}{l} * \\ \text{DATA} \end{array} \right\}$  [par]... [kommentar]  
–

- Beispiel:

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

## DD-Anweisung – DLM – 1

---

- Syntax

- DLM=delimiter

- Beispiel:

```
//JOBX JOB (3SLX510,000,00T NR0003), . . .
//STEP01 EXEC PGM=ZINS
//EINGABE DD *,DLM='??'
000119811982198319841985
000219861987198819891990
000319911992199319941995
//AUSGABE DD SYSOUT=A
??
//SYSPRINT DD DSN=XV10733.ispf.datei,DISP=SHR
```

- Beispiel 2:

```
//JOBX JOB (3SLX510,000,00T NR0003), . . .
//STEP01 EXEC PGM=ZINS
//EINGABE DD DATA,DLM=' &&&&'
000319911992199319941995
//AUSGABE DD SYSOUT=A
&&
//SYSPRINT DD DSN=XV10733.ispf.datei,DISP=SHR
```

# Datei-Beschreibung (1)

## DD-Anweisung – DLM – 3

---

| <b>DD-Anweisung</b> | <b>JES2</b> | <b>JES3</b> | <b>Bemerkung</b> |
|---------------------|-------------|-------------|------------------|
| DD *                | /* oder //  | /* oder //  | Normalfall       |
| DD *,DLM=xx         | xx oder //  | xx          | vermeiden!!      |
| DD DATA             | /*          | /*          |                  |
| DD DATA,DLM=xx      | xx          | xx          |                  |

# Datei-Beschreibung (1)

## DD-Anweisung – SYSOUT, HOLD

---

- Syntax

–  
– SYSOUT=  $\left\{ \begin{array}{l} \text{class} \\ ([\text{class}][, \text{INTRDR}][, \text{form-name}]) \\ * \end{array} \right\}$   
–

- 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=  
–

$$\left\{ \begin{array}{l} \text{nnn} \\ \\ \text{(nnn,(grp-wert[,grp-wert]...))} \end{array} \right\}$$

- Beispiel:

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

# Datei-Beschreibung (1)

## DD-Anweisung – SYSOUT, HOLD

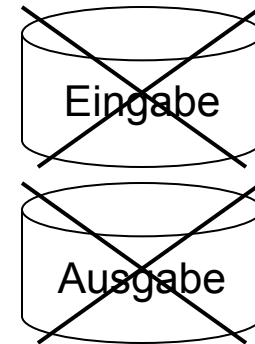
---

- 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”

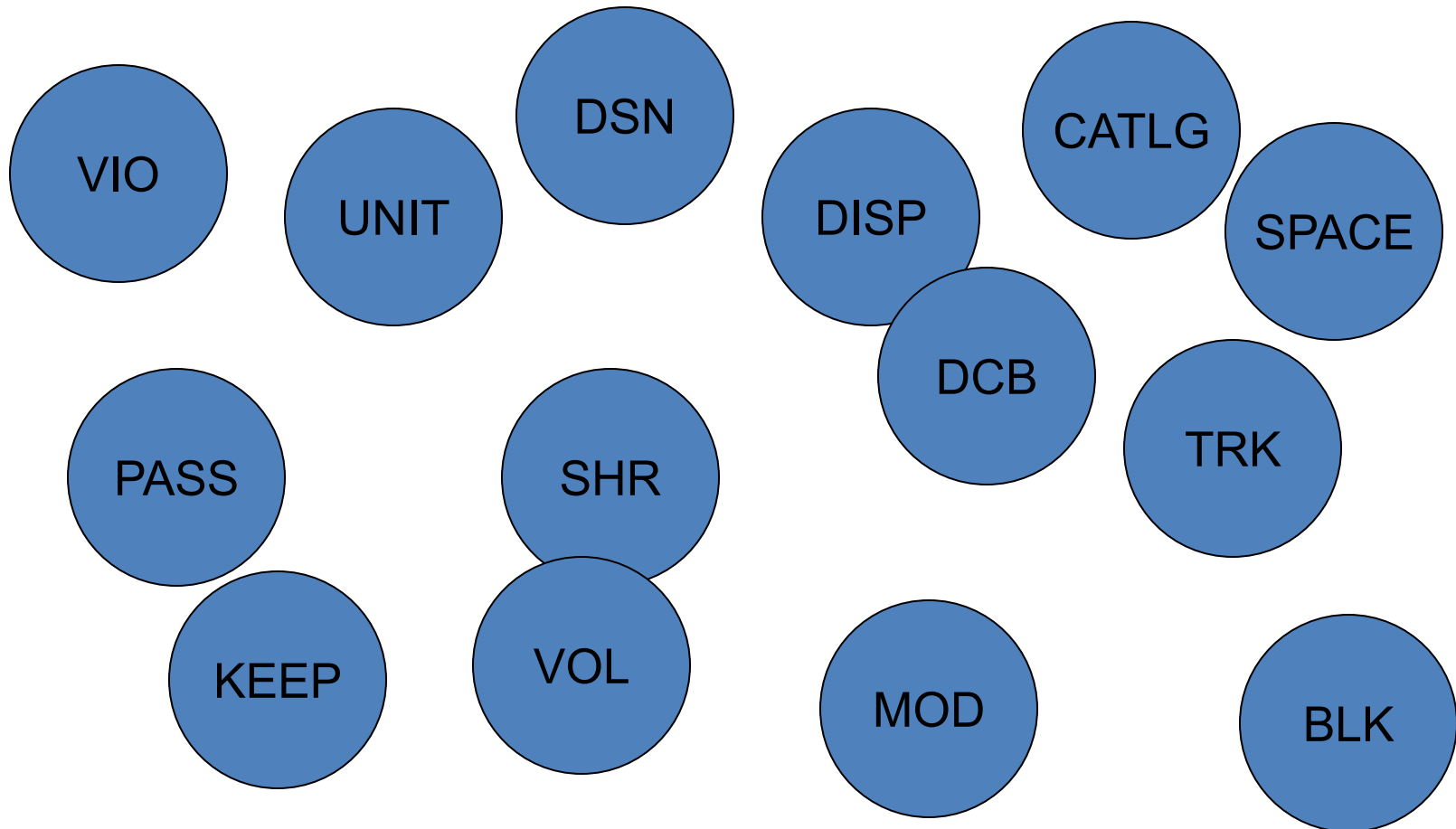


- 
- Einführung
  - Job-Beschreibung, Step-Beschreibung
  - Datei-Beschreibung (1)
  - • Datei-Beschreibung (2)
  - Standard- und Dienstprogramme - Überblick
  - Job-Steuerung, Step-Steuerung
  - Datei-Beschreibung (3)
  - Include-Gruppe, JCL-Prozedur
  - Diskussion und Austausch

# Datei-Beschreibung (2)

## Begriffe

---



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

## DD-Anweisung – DSNNAME

---

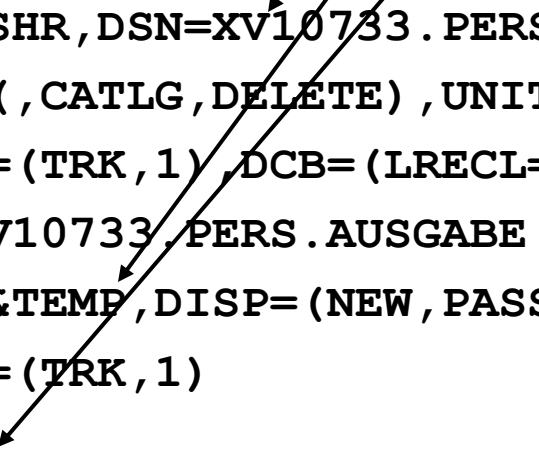
- Syntax

- DSN[AME]=dateiname

permanente Dateien  
temporäre Dateien  
Dummy Dateien

- Beispiel:

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



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

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

- **Anmerkungen zur Folgeseite:**

Fall 1: Der Datenbestand existiert bereits bei Jobbeginn oder wird während des Jobs mit KEEP oder CATLG behandelt.

Fall 2: Der Datenbestand 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,<br>MOD als NEW      | <weglassen>                                       | <weglassen>                                       | DELETE                           | DELETE                           |                                  |                                  |
| NEW,<br>MOD als NEW      | KEEP oder<br>DELETE oder<br>CATLG                 | <weglassen>                                       | <wie 2. Subparm>                 | <wie 2. Subparm>                 |                                  |                                  |
| NEW,<br>MOD als NEW      | KEEP oder<br>DELETE oder<br>CATLG                 | KEEP oder<br>DELETE oder<br>CATLG                 | <wie 2. Subparm>                 | <wie 3. Subparm>                 |                                  |                                  |
| NEW,<br>MOD als NEW      | PASS                                              | <weglassen>                                       | PASS                             | PASS                             | DELETE                           | DELETE                           |
| NEW,<br>MOD als NEW      | PASS                                              | KEEP oder<br>DELETE oder<br>CATLG                 | PASS                             | PASS                             |                                  | <wie 3. Subparm>                 |
| NEW,<br>(für temp)       | <weglassen>                                       | <beliebig>                                        | DELETE                           | DELETE                           |                                  |                                  |
| NEW,<br>(für temp)       | DELETE                                            | <beliebig>                                        | DELETE                           | DELETE                           |                                  | 1. Fall: KEEP<br>2. Fall: KEEP   |
| NEW,<br>(für temp)       | PASS                                              | <beliebig>                                        | PASS                             | DELETE                           | DELETE                           | 1. Fall: KEEP<br>2. Fall: KEEP   |
| OLD,SHR<br>MOD als OLD   | <weglassen>                                       | <weglassen>                                       | 1. Fall: KEEP<br>2. Fall: Delete | 1. Fall: KEEP<br>2. Fall: Delete |                                  |                                  |
| OLD,SHR<br>MOD als OLD   | KEEP oder<br>DELETE oder<br>CATLG oder<br>UNCATLG | <weglassen>                                       | <wie 2. Subparm>                 | <wie 2. Subparm>                 |                                  |                                  |
| OLD,SHR<br>MOD als OLD   | KEEP oder<br>DELETE oder<br>CATLG oder<br>UNCATLG | KEEP oder<br>DELETE oder<br>CATLG oder<br>UNCATLG | <wie 2. Subparm>                 | <wie 3. Subparm>                 |                                  |                                  |
| OLD,SHR<br>MOD als OLD   | PASS                                              | <weglassen>                                       | PASS                             | PASS                             | 1. Fall: KEEP<br>2. Fall: Delete | 1. Fall: KEEP<br>2. Fall: Delete |
| OLD,SHR<br>MOD als OLD   | PASS                                              | KEEP oder<br>DELETE oder<br>CATLG oder<br>UNCATLG | PASS                             | PASS                             | 1. Fall: KEEP<br>2. Fall: Delete | <wie 3. Subparm>                 |

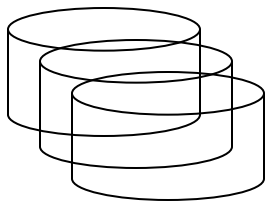
# Datei-Beschreibung (2)

## Datenbestände – UNIT / VOL

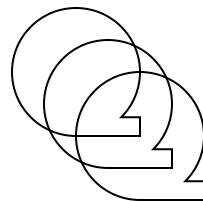
---

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

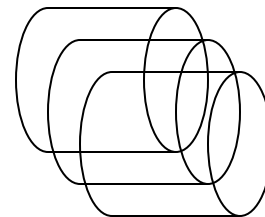
SYSDA



REEL



TAPE



etc.



# Datei-Beschreibung (2)

## Datenbestände – SPACE (ohne SMS)

---

- Syntax

–  
– SPACE=(  
–

$$\left. \begin{array}{l} \text{blklen,} \\ \text{TRK, (prim[,sec][,dir)][,RLSE]} \\ \text{CYL,} \end{array} \right\}$$

- Beispiel:

```
//SYSIN DD DISP=SHR,DSN=XV10733.PERS.DATEN
//SYSPRINT DD DISP=(,CATLG,DELETE),UNIT=SYSDA,
// SPACE=(TRK,1),DCB=(LRECL=80,RECFM=FB),
// DSN=XV10733.PERS.AUSGABE
//WORKFILE DD DSN=&&TEMP,DISP=(NEW,PASS),UNIT=SYSDA,
// SPACE=(TRK,(15,30))
```

- Syntax

- DCB=(subparm[,subparm] . . .)

- Data Control Block

- Beispiel:

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

- Reihenfolge ist zu beachten
  - aus Programm
  - aus DD-Anweisung
  - aus Definition der Datei auf Platte/Band
  
- Anmerkung:
  - Es gibt keine Faustregel, was wo angegeben werden soll; eine genaue Kenntnis der Anwendung ist erforderlich.

- 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     | -      |



## 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

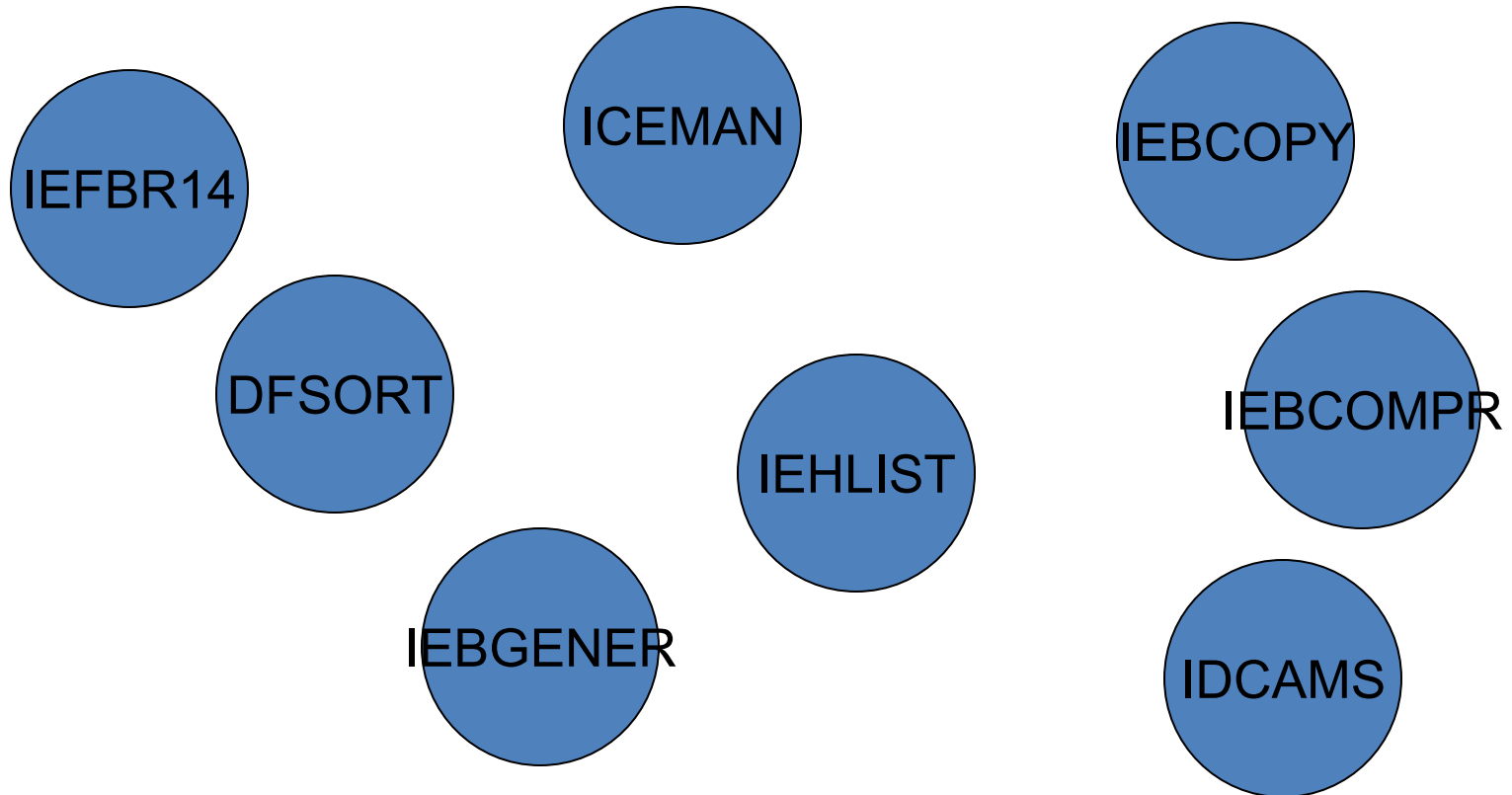


- 
- Einführung
  - Job-Beschreibung, Step-Beschreibung
  - Datei-Beschreibung (1)
  - Datei-Beschreibung (2)
  - ➔ • Standard- und Dienstprogramme - Überblick
  - Job-Steuerung, Step-Steuerung
  - Datei-Beschreibung (3)
  - Include-Gruppe, JCL-Prozedur
  - Diskussion und Austausch



## Begriffe

---



# Datei-Beschreibung (2)

## Übung(en)

---

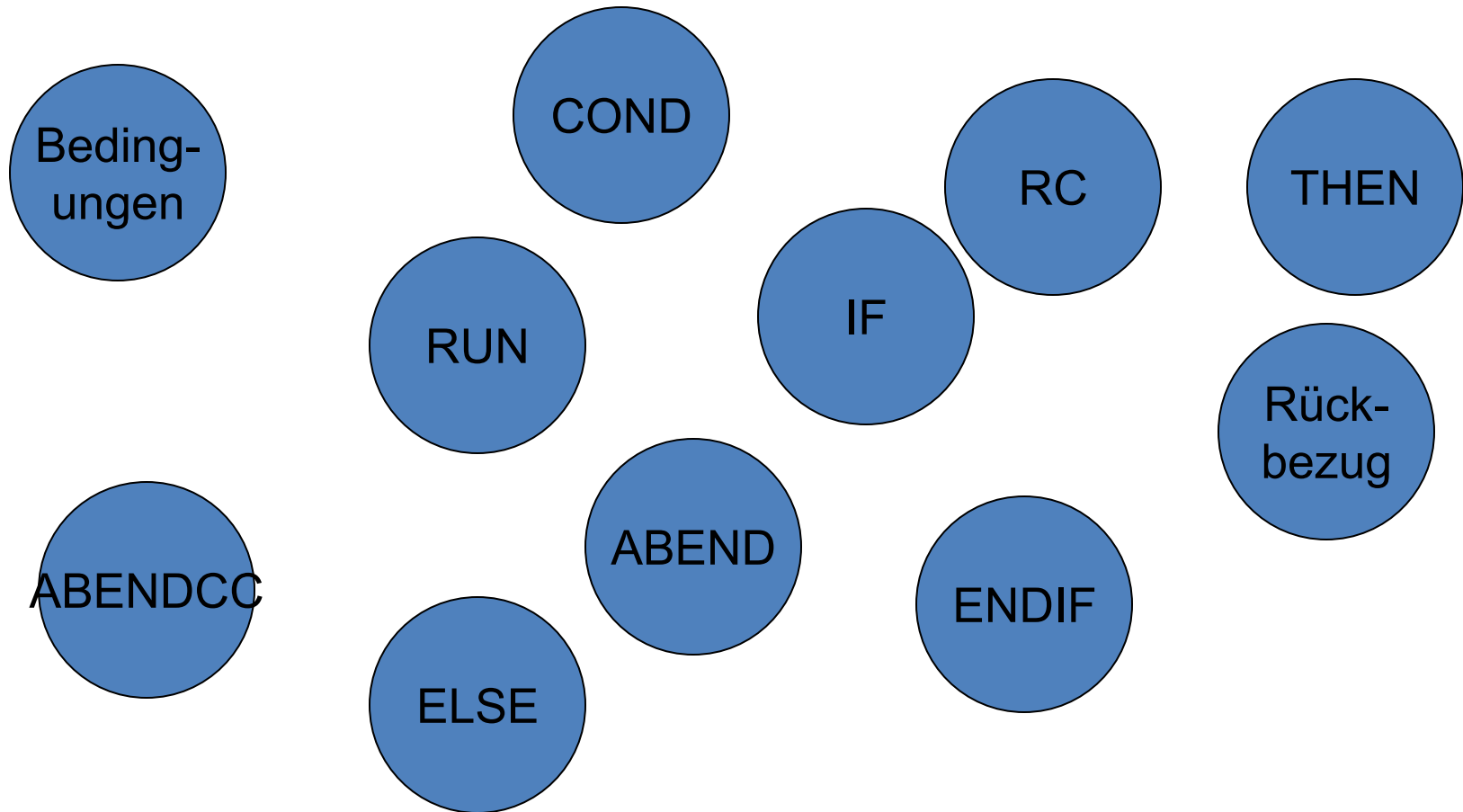
- Kapitel 5.1: Anlegen DSN mit IEFBR14
- Kapitel 5.2: Editieren Datei im EDIT
- Kapitel 5.3: Kopieren Datei mit IEBGENER
- Kapitel 5.4: Löschen DSN mit IEFBR14



- 
- 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

## Begriffe

---



- 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

- 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)

## 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))
```

## 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)
```



## 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                   |

## 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

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

- 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 | <= |

## 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 | ^= |

## 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



- Test, ob bestimmter Step ausgeführt wurde
- Syntax
  - step.RUN=TRUE oder step.RUN
  - step.RUN=FALSE oder NOT step.RUN oder ^step.RUN

- Unterdrückte Step 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

- 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

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

- Syntax
  - DSNNAME=\*[.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)
```




## 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



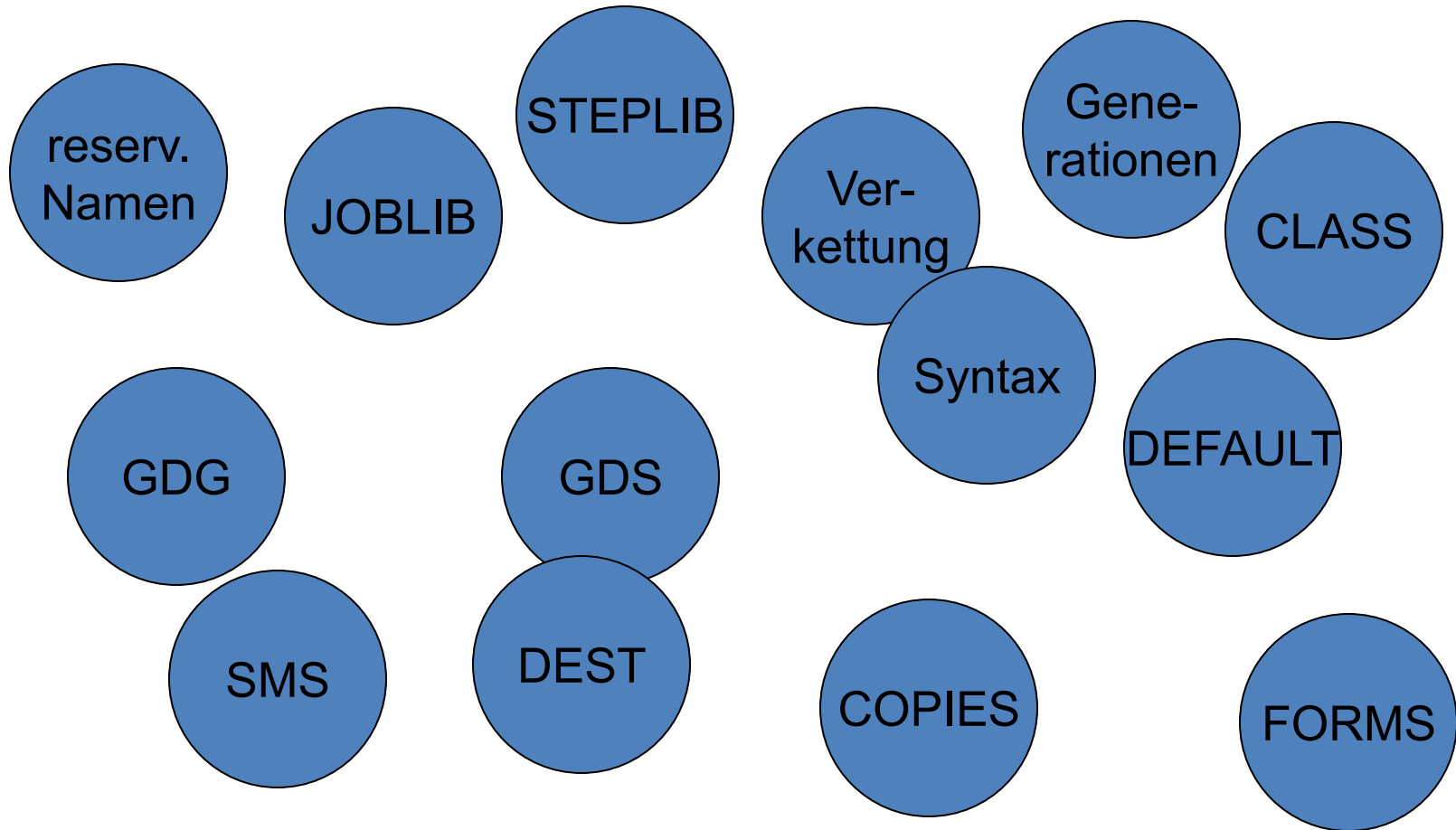
- 
- Einführung
  - Job-Beschreibung, Step-Beschreibung
  - Datei-Beschreibung (1)
  - Datei-Beschreibung (2)
  - Standard- und Dienstprogramme - Überblick
  - Job-Steuerung, Step-Steuerung
  -  • Datei-Beschreibung (3)
  - Include-Gruppe, JCL-Prozedur
  - Diskussion und Austausch



# Datei-Beschreibung (3)

## Begriffe




---



# Datei-Beschreibung (3)

## reservierte DD-Namen – Überblick

---

- SYSMDUMP - Dump
-  • SYSABEND, SYSUDUMP - formatierter Dump
- SYSCHK, SYSCHKEOV - Checkpointing
-  • JOBCAT, STEP CAT - Katalogangabe
-  • JOBLIB, STEPLIB - Ladebibliotheken

# Datei-Beschreibung (3)

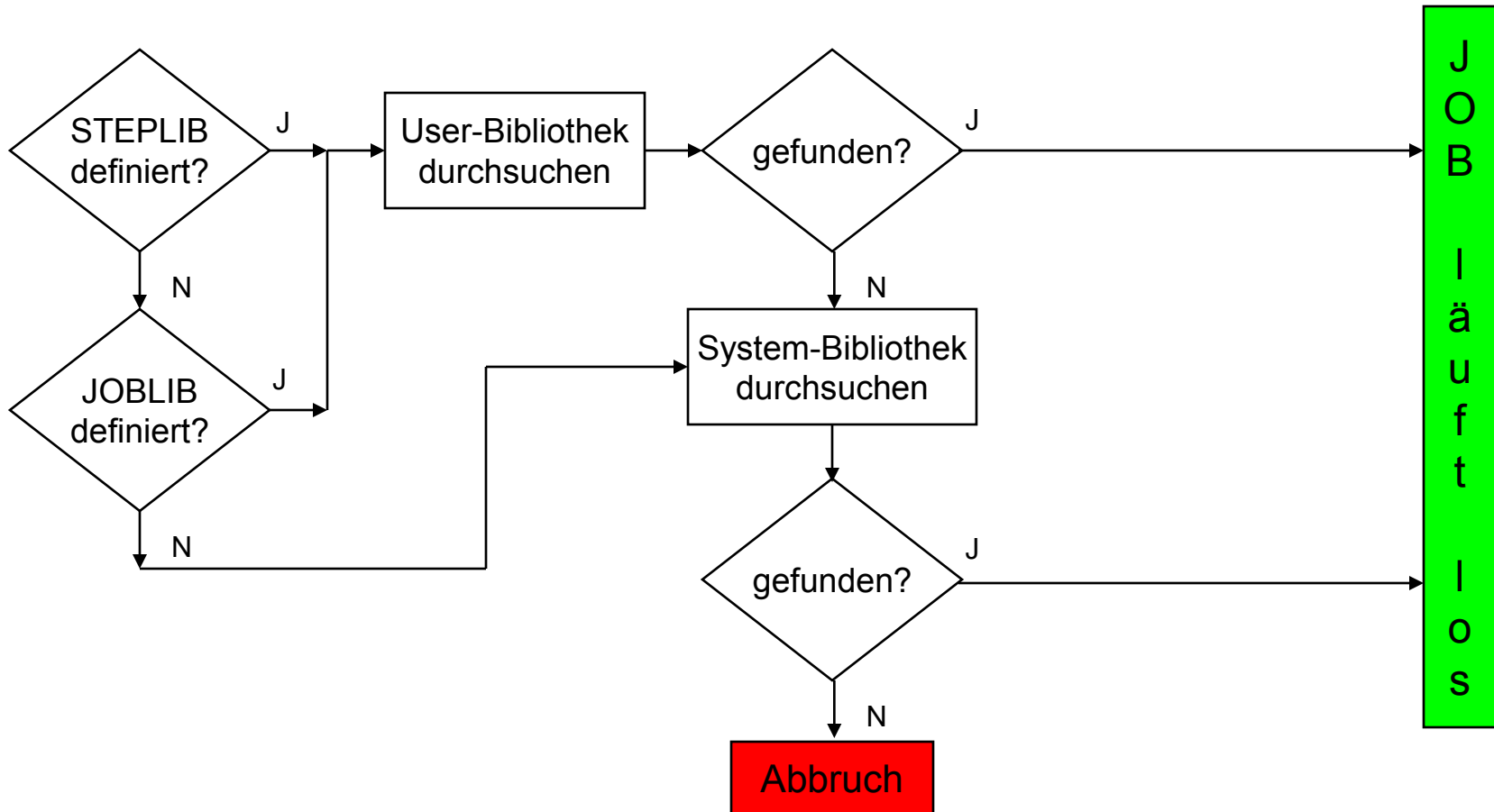
## reservierte DD-Namen – JOBLIB, STEPLIB

---

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

# Datei-Beschreibung (3)

## reservierte DD-Namen – JOBLIB, STEPLIB (PAP)



- 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.

- 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

- 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

- Beispiel:

```
//XV10733 JOB etc.
//*----- Anlegene GDG
//DEFGDG EXEC PGM=IDCAMS
//SYSPRINT DD SYSOUT=*
//SYSIN DD *
 DEFINE GDG (NAME (XV10733.LAGER.BESTAND) -
 LIMIT(12) SCRATCH)
```



## 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



## 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



## 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



- Syntax
  - //name OUTPUT parameter[,parameter]
  
- Parameter
  - DEFAULT=(YES|NO)
  - CLASS=(class|\*)
  - FORMS=(formname)
  - COPIES=(nnn)
  - DEST=ziel

- 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

# Datei-Beschreibung (3)

## OUTPUT-Anweisung – weitere Parameter

---

- TITLE
- NAME
- ADDRESS
- DEPT
- BUILDING
- ROOM
- etc.



# Datei-Beschreibung (3)

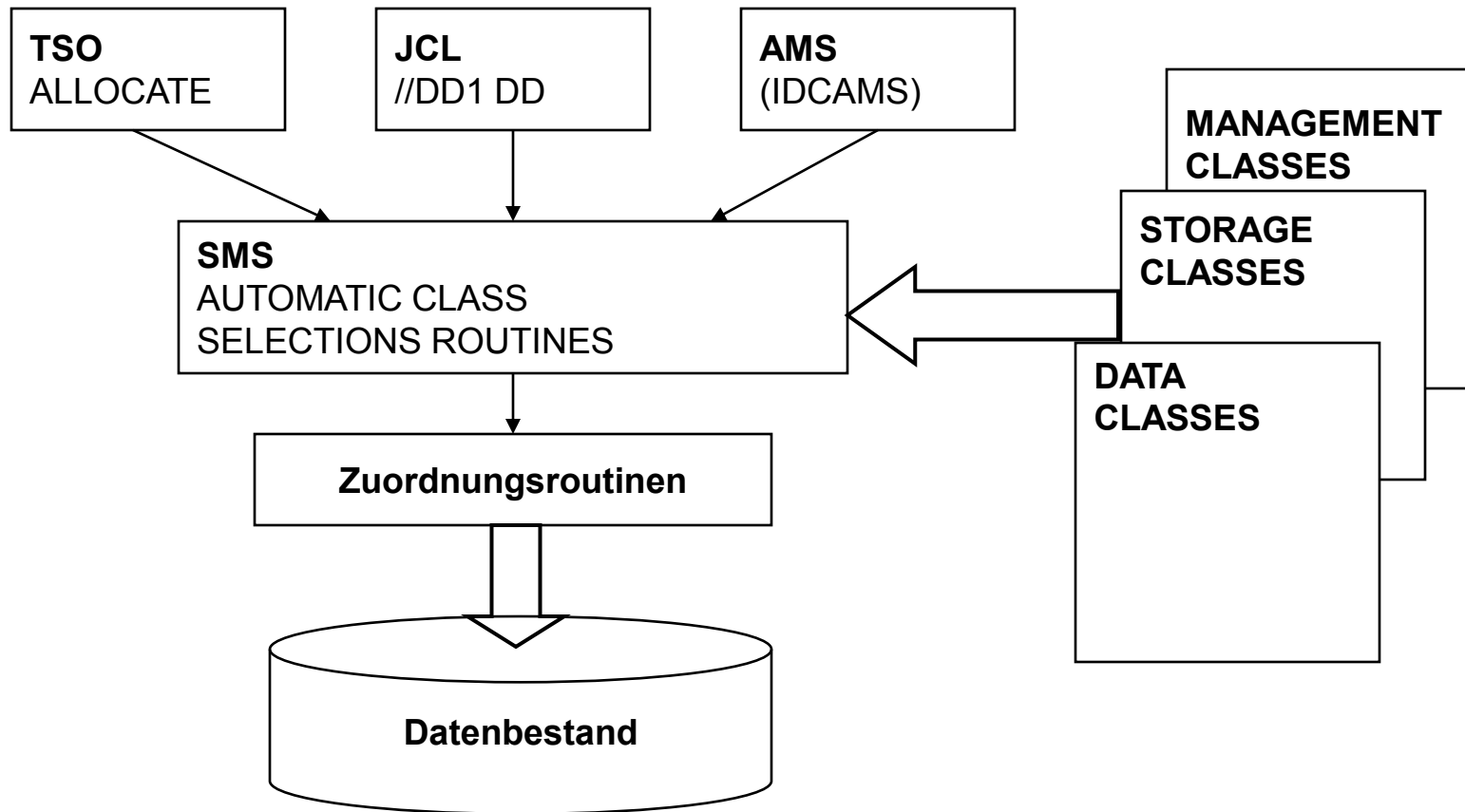
## OUTPUT-Anweisung – only to confuse the russians . . .

```
//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

---

| <b>DATA CLASS Attribute (Auswahl)</b> |               |              |              |               |               |               |              |               |
|---------------------------------------|---------------|--------------|--------------|---------------|---------------|---------------|--------------|---------------|
| <b>Name</b>                           | <b>RECORG</b> | <b>RECFM</b> | <b>LRECL</b> | <b>KEYLEN</b> | <b>KEYOFF</b> | <b>AVGREC</b> | <b>VALUE</b> | <b>CISIZE</b> |
| 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, RECOG, LRECL etc.

---

- RECFM: nicht VSAM
- RECOG: 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

---

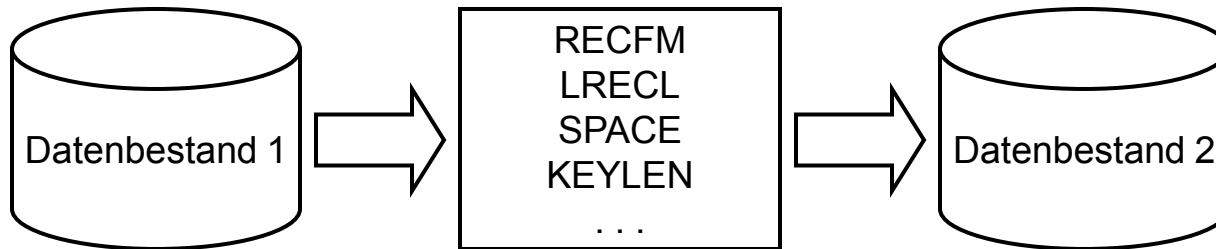
| <b>PO / PDS</b>                                                   | <b>PDSE / LIBRARY</b>                                 |
|-------------------------------------------------------------------|-------------------------------------------------------|
| Kann vom SMS verwaltet werden                                     | 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:                                                         |                                                       |
| <code>//AUSG DD DSN=XV10733.JCL.CNTL,DISP=(,CATLG,DELETE),</code> |                                                       |
| <code>// DATACLAS=DATAPO,DSNTYPE=LIBRARY</code>                   |                                                       |

# Datei-Beschreibung (3)

## SMS-Speicherverwaltung – LIKE, REFDD

---

- SYNTAX
  - LIKE=datasetname



- Beispiel:

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

# Datei-Beschreibung (3)

## SMS-Speicherverwaltung – STORCLAS

---

| STORAGE CLASS - Attribute (Auswahl) |                      |                       |                       |                       |              |                     |
|-------------------------------------|----------------------|-----------------------|-----------------------|-----------------------|--------------|---------------------|
| Name                                | Dir Response<br>µsec | Use (Read /<br>Write) | Seq. Response<br>µsec | Use (Read /<br>Write) | Availability | Guaranteed<br>Space |
| STANDARD                            | -                    | -                     | 10                    | -                     | STANDARD     | NO                  |
| KRITISCH                            | 10                   | -                     | 10                    | -                     | STANDARD     | NO                  |
| DBSTD                               | -                    | -                     | -                     | -                     | STANDARD     | YES                 |
| DBFAST                              | 10                   | R                     | 10                    | R                     | CONTINUOUS   | 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=DATAF'S,STORCLAS=STANDARD
```



# Datei-Beschreibung (3)

## SMS-Speicherverwaltung – MGMTCLAS

MANAGEMENT CLASS - Attribute (Auswahl)

| Name     | Expire Non-Usage | Expire Date / Days | Max. Retention 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=DATAF'S,STORCLAS=STANDARD,
// MGMTCLAS=STANDARD
```



# Datei-Beschreibung (3)

## Übung(en)

---

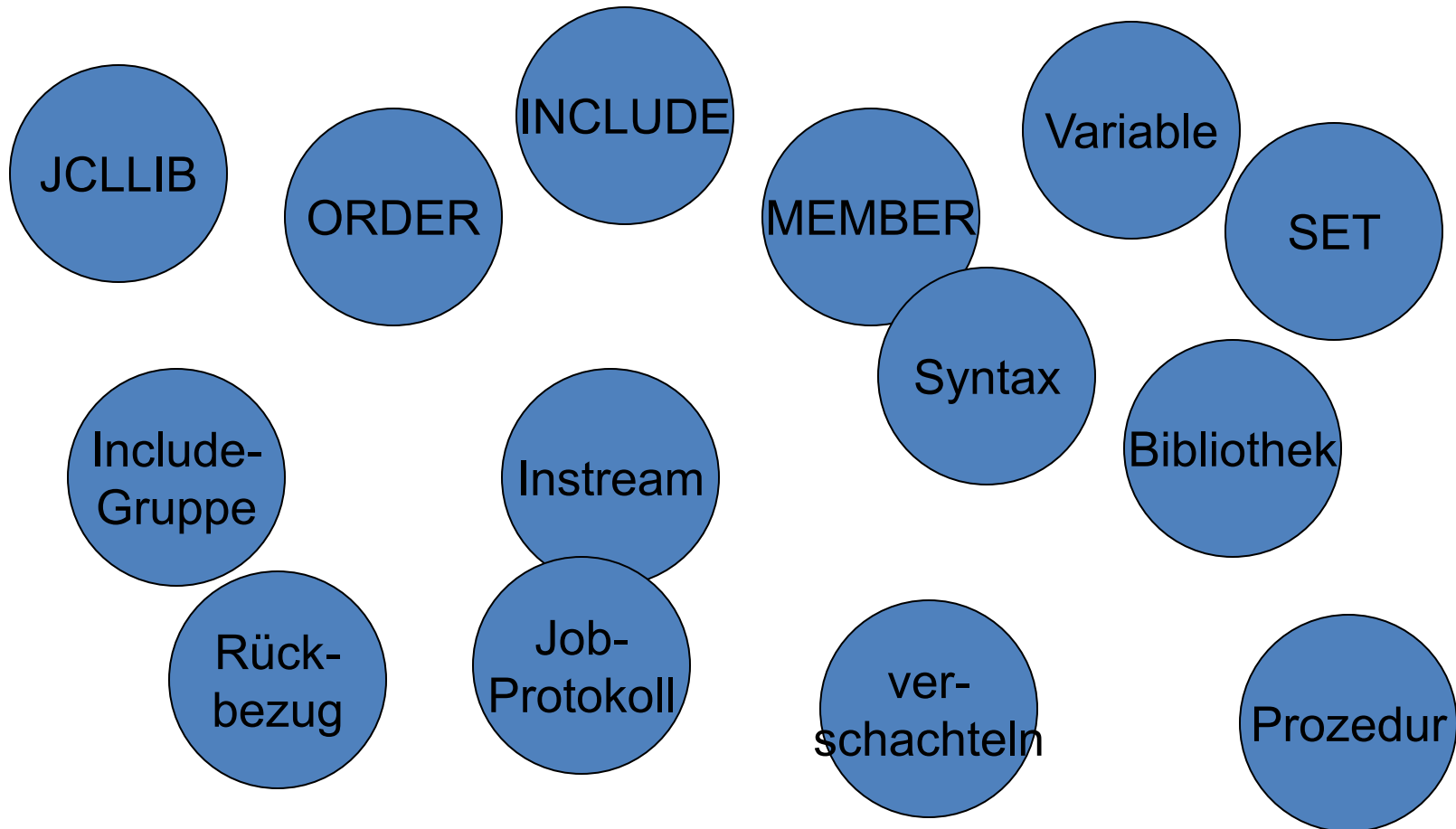
- Kapitel 7.10: welche STORCLAS in Firma
- Kapitel 7.11: welche MGMTCLAS in Firma



- 
- 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

## Begriffe

---



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

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

## 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-Anweisung – Überblick

---

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



## 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-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
```

## 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

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

## 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
```

## 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.



## 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)

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





- 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.



## 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
```

- 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
```

## 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



- 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

- 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

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



- 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





- 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

## 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
```



## 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

- 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

- siehe Firmenbibliothek



## 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



- 
- 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

