Kapitel 5 – Job Control Language

Einen wesentlichen Bestandteil in z/OS stellt die Job Control Language dar. Es ist nicht sinnvoll, Alles und Jedes unter TSO zu verarbeiten – nur weil es geht. Das Problem ist die zunehmende Intelligenz der Workload Manager. Wird unter TSO eine Verarbeitung gestartet, bei der sehr viele Datensätze gelesen/geschrieben (womöglich gar sortiert) werden, wird in aller Regel weder Bildschirm noch Tastatur in diesen Vorgang eingebunden sein. Dies ist ein Indiz, dass hier ein Missbrauch stattfindet, welcher durch Reduzierung der Systemservices geahndet wird. Im Klartext heißt das: Je länger eine Verarbeitung dauert, desto langsamer wird sie.

Aus diesem Grund ist es empfehlenswert, Verarbeitungen, welche ein großes Datenvolumen zu bewältigen haben, nicht unter TSO, sondern im Hintergrund ablaufen zu lassen. Das geht selbstverständlich auch mit REXX-Programmen – bis zu einem gewissen Grad sogar dann, wenn sie ISPF-Services aufrufen.

Wir wollen zunächst einige Randinformationen über das Spoolsystem des z/OS betrachten, ohne welches eine Batchverarbeitung nicht möglich wäre.

Das Spoolsystem

Die Verwaltung der Arbeitsaufträge (Jobs) wird nicht vom z/OS selbst, sondern von den Job-Entry-Subsystemen JES2 oderJES3 übernommen. Aus diesem Grund ist ein OS/390 ohne Spoolsystem entgegen vielen anderen Betriebssystemen nicht arbeitsfähig.

Ein Arbeitsauftrag an die DV-Anlage wird durch Steueranweisungen in einer eigens dafür bestimmten Sprache (JCL = Job-Control-Language) beschrieben.

Um einen reibungslosen Verarbeitungsablauf zu gewährleisten, trägt das Spoolsystem im Einzelnen bei:

- Einlesen der Arbeitsaufträge (Jobs)
- Verwalten der Auftragswarteschlangen
- Vorbereitende Arbeiten und Übersetzen von Job-Control-Anweisungen
- Bereitstellen von Prozeduren aus der SYS1.PROCLIB
- Verwalten von Spool-Datenbeständen
- Durchführung der Systemausgaben.

Internal Reader

Früher wurden Jobs in Form von 80-stelligen Lochkarten über einen Kartenleser (Reader) eingelesen. Als Ausgabemedium standen dem Spoolsystem Stanzer (Puncher) und Drucker (Printer) zur Verfügung.

Heute werden mit Hilfe des TSO-Betriebs über den ISPF-Editor 80-stellige JCL-Anweisungen über den INTERNAL-READER (Programm, das den Kartenleser simuliert) eingelesen. INTERNAL-READER können auch für Kassetten- oder Platteneinheiten gestartet werden, falls Jobs von Kassetten oder Platten gelesen werden sollen.

External Writer

Die Durchführung der Systemausgabe (SYSOUT) erfolgt standardgemäß über Drucker oder Stanzer. Soll die Ausgabe auf ein anderes Ausgabemedium (Magnetplatte, Band, oder ähnliches) erfolgen, geschieht dies über den External Writer.

Prioritäten

JES verwaltet Jobs in einer Warteschlange (Job Queue). Alle eingelesenen Jobs werden dabei in einer Spooldatei zwischengespeichert (Spooling). Wie die Jobs der Reihe nach abgearbeitet werden, kann durch Prioritätsangaben in der Job-Anweisung (PRTY=zahl) geregelt werden. Die maximale Prioritätsangabe ist 13, die niedrigste ist 0. Bei gleichen Prioritätsangaben entscheidet JES im Allgemeinen nach dem Motto FIFO (First In First Out), oder "Wer als erster kommt, malt zuerst" die Reihenfolge der Auswahl.

Job Auswahl

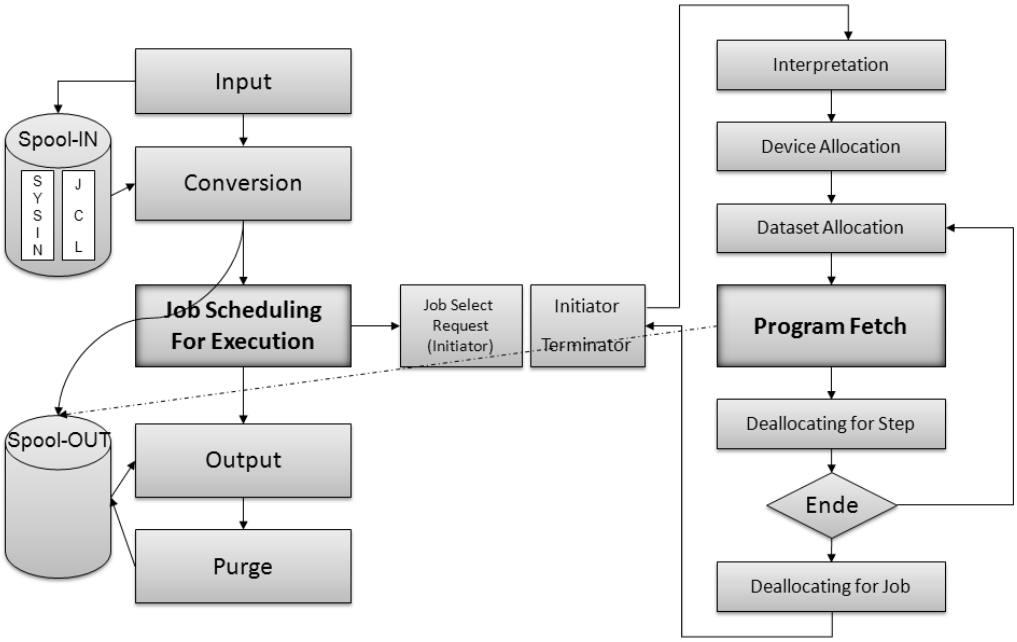
Die Einleitung zur Ausführung von Jobs geschieht durch den Initiator. Pro Initiator kann immer nur ein einziger Job laufen. Theoretisch können 1000 Initiator definiert werden (Default = 3 Initiator). Der Initiator fragt ständig nach, ob Jobs zur Ausführung anstehen. Er wählt zu seinen Job-Klassen die zugehörigen Jobs aus, ordnet die notwendigen Datenbestände, Datenträger und E/A-Einheiten zu und führt das aufgerufene Programm aus.

JES2 oder JES3?

JES2 wurde ursprünglich für ein Rechenzentrum mit einem großen Rechner konzipiert. Eine Mehrrechnerlösung wird heute über MAS (Multi Access Spool) realisiert.

JES3 wurde für Mehranlagenbetrieb geschaffen, wobei die Erstellung von Job-Netzen ermöglicht wurde.

Schaubild Durchlaufphasen JES2



JES-Durchlaufphasen

JES2 durchläuft die Phasen

- Input
- Conversion
- Job Scheduling for Execution
- Output
- Purge

JES3 durchläuft zwischen der Input-Phase und der Scheduling-Phase drei weitere Phasen. Ein großer Vorteil liegt dabei in der Generalized Main Scheduling Phase, welche Dateien zuweist. Dieses Manko von JES2 wird durch SMS wettgemacht.

- Conversion/Interpretation
- Main Device Scheduling
- Generalized Main Scheduling

Input-Phase

Hier erfolgt zunächst das Einlesen der Jobs. Im Allgemeinen wird der Vorgang des Lochkartenlesens durch den INTERNAL READER simuliert. Eine eindeutige Jobidentifikationsnummer wird zugeordnet. Der Job wird in den SPOOL-IN-Bereichen zwischengespeichert, wobei eine Trennung zwischen reiner JCL und Instream-Datenbeständen erfolgt. Die JCL-Sätze werden zur Formalüberprüfung an die

Conversion-Phase

übergeben. Anschließend wird die im Job über Prozeduren aufgerufene JCL eingearbeitet. Alle JCL-Anweisungen werden in Kontrollblöcke (SWA-Blocks) übersetzt und wieder in der Spooldatei abgelegt.

Interpretation-Phase

Diese Phase existiert nur bei JES3 und stellt eine Erweiterung der Conversion-Phase dar. Im JES2 wird diese Phase erst in der Execution-Phase durch den Initiator wahrgenommen. Bereits hier wird festgestellt, welche Datenbestände, Datenträger und E/A-Einheiten benötigt werden. Es werden die Steuerblöcke für den Job-Scheduler gebildet (SWA-Kontrollblöcke) und die Zwischenspeicherung dieser Blöcke in die Spooldatei vorgenommen.

Main Device Scheduling

Alle Datenbestände, Datenträger und E/A-Einheiten werden bereitgestellt. Eventuelle Aufforderung an die Operator zur Montage von Datenträgern ergehen. Es wird überwacht, dass eine Datei nicht gleichzeitig mehreren Jobs zugeteilt wird. Der Job wird weitergegeben, wenn alle Anforderungen erfüllt sind.

Generalized Main Scheduling

Hier wird die Jobverteilung optimiert. Die Verteilungsregeln werden je nach Art der Jobs (E/A-intensiv oder rechenintensiv) vom Systemprogrammierer festgelegt. Daraus ergeben sich dann bestimmte Job-Scheduling-Algorithmen.

Execution Phase

Die Execution-Phase besteht aus mehreren Aktivitäten des Initiators (eigener Adressraum pro Initiator). Der Initiator fordert vom JES über die CSA einen Job an. Entsprechend der Jobklasse und der Auswahlpriorität wird der Job aus der Job-Queue ausgewählt.

Interpreter

Diese Aufgabe wird an dieser Stelle nur im JES2 benötigt (Conversion/Interpretation-Phase bei JES3). Es werden Kontrollblöcke für die Dauer eines Jobs in der SWA angelegt.

Allocator

Auch diese Funktion betrifft an dieser Stelle nur JES2 (Main Device Scheduling bei JES3). Die Dateizuweisung wird durchgeführt. Dabei wird geprüft, ob eine Datei bereits von einem anderen Job angesprochen wurde. Je nachdem was in der JCL-Anweisung (DISP-Parameter) angegeben wurde, wird eine Datei nur von einem Job exklusiv benutzt oder kann von mehreren Jobs parallel angesprochen werden. Bei exklusiver Angabe bekommt der Operator die Meldung "Job Waiting for Datasets".

Zusätzlich wird beim Allocator die Gerätezuweisung der benötigten E/A-Einheiten getroffen. Bei JES2 findet die Zuordnung immer stepweise statt. Sind benötigte Datenträger nicht montiert, wird der Operator aufgefordert dies zu tun. Bleibt der Job in dieser Phase hängen, weil momentan nicht genügend E/A-Geräte zur Verfügung stehen, erhält der Operator über entsprechende Kommandos die Möglichkeit, den Job in Wartezustand zu versetzen, oder abzubrechen.

Program Fetch/Program Execution

Die Programme der einzelnen Jobsteps werden geladen und ausgeführt.

Deallocating Datasets und Devices for Jobsteps.

Ist ein Step beendet, werden alle Datenbestände, Datenträger und E/A-Einheiten, die in darauffolgenden Steps nicht mehr gebraucht werden, freigegeben. Bei JES2 findet eine erneute Zuordnung von Dateien, Datenträgern und E/A-Einheiten bei jedem weiteren Step statt. Bei JES3 wird der nächste Step ausgeführt.

Deallocating Datasets und Devices for Job

Nach Beendigung des Jobs (alle Steps) werden alle Dateien, Datenträger und E/A-Einheiten freigegeben.

Job Termination Request

Das Ende eines Jobs wird JES mitgeteilt. Der Initiator übergibt die Steuerung über den Job an das JES zurück.

Output-Phase

Die Ausgabe erfolgt normalerweise auf einen Drucker. Soll die Ausgabe über Bänder, Kassetten oder Plattenspeicher erfolgen, muss dies über den EXTERNAL WRITER erfolgen.

Die Ausgabe kann aber auch an lokale oder andere über Leitung verbundene E/A-Geräte ausgegeben werden (RJE = Remote Job Entry) oder sogar an andere Rechner in verschiedenen Orten mit unterschiedlichen Betriebssystemen (NJE = Network Job Entry).Es können sowohl Jobs als auch Output in jede Richtung geschickt werden. Heute wird dieser Weg häufig als Datentransfer missbraucht. Voraussetzung ist JES2 oder JES3 als Job-Management-System.

Purge-Phase

Alle Bereiche, die von dem betreffenden Job innerhalb der Spooldatei belegt wurden, werden freigegeben. Die Eintragungen der beendeten Jobs werden aus der Job-Output-Queue gelöscht. Der Operator wird über das Ende der Purge-Phase informiert.

Steueranweisungs-Typen

Generell gilt: Alle JCL-Anweisungen müssen in Upper Case kodiert sein. In der Job Control Language gibt es eine Reihe unterschiedlicher Anweisungstypen. Diese sind:

Anweisungstyp	Bedeutung
JOB	Gibt dem Arbeitsauftrag einen Namen
EXEC	Ruft ein Programm oder eine JCL-Prozedur auf
DD	Erklärt das Datenuniversum für ein Programm
NULL	Früher zwingend als ENDE, wird heute nicht mehr gebraucht
JCLLIB-DD	Zuweisung privater Procedure-Bibliotheken
SET	Zuweisung von Variablen
INCLUDE	Import von JCL vergleichbar zu File Tailoring
JES-Command	Ausführung von JES-Commands via JCL. Ist nur wenigen Mitarbeitern erlaubt.
JES-Anweisungen	Routing, Drucker-/Checkpoint-Definitionen
PROC	Procedure-Beginn-Anweisung
PEND	Procedure-END-Anweisung

Viele Verarbeitungen lassen sich jedoch mit zwei dieser Typen erledigen. Die Minimalangaben sind die Anweisung EXEC und deren dazugehörige DD-Statements.

Reihenfolge der Anweisungen

```
//name     JOB .....
//         PROC ....
//         ....
//         PEND
//JOBLIB   DD ......
//name     EXEC ....
//STEPLIB  DD ......
//name     DD ......
//name     DD DATA,DLM=##
daten
##
//name     DD ......
//
```

Aufbau und Einteilung der Anweisungen

Die JCL-Statements unterteilen sich in verschiedene Felder. Obligatorisch und Erkennungszeichen der JCL-Befehle sind Schrägstriche(//) auf den Spalten 1 und 2.

Abgelegt werden die Statements in einer Datei fester Länge von 80 Byte (ein Relikt aus der Zeit der Lochkarten).

Es gibt insgesamt 4 Felder. NAME, TYP, OPERAND(EN), KOMMENTAR. Zwischen den einzelnen Feldern muss mindestens eine Leerstelle sein.

Das Namensfeld muss auf Spalte 3 beginnen und darf maximal 8 Byte (alphanumerisch) lang sein. Die erste Stelle muss ein Buchstabe sein.

In der JOB-Anweisung muss das Namensfeld angegeben sein.

In der EXEC-Anweisung muss das Namensfeld zwingend angegeben werden, wenn an anderer Stelle auf diesen Step Bezug genommen werden soll. Der besseren Transparenz wegen sollte es aber immer versorgt werden.

In der DD-Anweisung wird der symbolische Dateiname angegeben, welcher im Programm benutzt wird. Der Name kann nur bei Dateiverkettung ab der zweiten Stelle weggelassen werden.

Im Operandenfeld stehen Anweisungen für die Operation. Wenn die Operanden nicht in einer Zeile Platz finden, kann in der nächsten Zeile fortgefahren werden. Die Trennung kann nur nach vollständiger Angabe eines Operanden erfolgen und wird durch Abschluss einer Zeile mit Komma(,) durchgeführt.

Die Fortsetzungszeile beginnt mit //. Die Operanden müssen dann zwischen den Spalten 4 und 16 weiter geschrieben werden.

Das Kommentarfeld muss durch eine Leerstelle vom Operandenfeld getrennt werden.

In der Fortsetzungsspalte(72) kann ein Zeichen ungleich BLANK angegeben werden.

Bedeutung der Sonderzeichen

Name	Zeichen	Bedeutung
Alphabetic Numeric At Hash Dollar Sign	A-Z 0-9 @ # $	Namen der Anweisungen, Stelle 1 muss ein Buchstabe sein
Comma	,	Trennzeichen zwischen Parametern
Period	.	Trennung der Dateinamensteile
Slash	/	Merkmal für JCL auf den Stellen 1 und 2

Name	Zeichen	Bedeutung
Apostrophe	'	Schließt alle Angaben eines Parameters ein und erlaubt dabei Sonderzeichen
Parenthesis	()	Eingrenzung von Sub-Parametern
Asterisk	*	Referenzangaben, meist Rückbezugnahmen
Plus Hyphen	+ -	Kennzeichnet die relative Generation einer Generation Data Group
Equal	=	Wertzuweisung für Schlüsselworte
Blank		Feldbegrenzer

Werden andere als die genannten oder generell erlaubte Zeichen am falschen Ort kodiert, führt dies immer zum Abbruch von JCL.

Es ist empfehlenswert, sich die englischen Namen der Sonderzeichen einzuprägen. In einem Fehlerprotokoll steht womöglich "Incorrect use of parenthesis" aber mit Sicherheit niemals "Sie haben die Klammern falsch gesetzt".

Die Job-Anweisung

```
//jobname JOB operanden
```

Jobname

Mit dem Jobnamen wird ein Auftrag im System definiert. Die Angabe ist zwingend.

Positionsparameter

Positionsparameter müssen immer vor den Schlüsselwortparametern geschrieben werden. Die Reihenfolge der Stellungsparameter ist einzuhalten. Zwei Angaben sind möglich. Wird die erste Angabe weggelassen, muss ein Komma(,) gesetzt werden.

account-information

Der erste Stellungsparameter ist die Abrechnungsinformation. Die Angaben werden zwischen runde Klammern() gesetzt. Wird mehr als eine Angabe gemacht, sind die einzelnen Angaben innerhalb der Klammern durch Komma zu trennen.

programmierer-name

ist der zweite Stellungsparameter. Hier kann eine beliebige Eintragung bis zu 20 Byte Länge angegeben werden. Um Probleme bei der Verwendung von Sonderzeichen auszuschließen, sollte dieser Eintrag zwischen Hochkommata gesetzt werden.

Beispiel

```
//XWITT1 JOB (11199002),'AUSBILDUNG', Stellungsparameter
//          CLASS=...                 Schlüsselwort
```

Schlüsselwort-Parameter

Die Reihenfolge der Schlüsselwortparameter ist beliebig, da sie auf Grund ihres Namens erkannt werden. Wird ein Parameter nicht angegeben, so ist, soweit definiert, der Standardwert gültig.

Mögliche Angaben (Defaultwerte sind angegeben. Diese sind aber abhängig von der jeweiligen Installation und nicht zwingend so):

Schlüsselwort	Bedeutung
CLASS	Verarbeitungsklasse, in welcher der JOB läuft
COND	Regelt die Ablaufbedingung(en) unter denen eine Verarbeitung stattfinden soll (oder nicht).
MSGCLASS	Benennt die Ausgabeklasse, in der das Verarbeitungsprotokoll abgelegt wird.
MSGLEVEL	Bestimmt, wie aussagekräftig und umfangreich das Verarbeitungsprotokoll ist.
NOTIFY	Benennt einen Benutzer, der bei Jobende benachrichtigt wird.
REGION	Bestimmt die Adressraumgröße für die Verarbeitung
TIME	Kann eine Laufzeitbegrenzung durchführen
ADDRSPC	Entscheidet, ob Pageing für die Verarbeitung zulässig ist
PERFORM	Bestimmt eine Performance Gruppe (wird heute meist durch WLM-Angaben geregelt)
PRTY	Bestimmt die Auswahlpriorität im Spoolsystem

Die EXEC-Anweisung

Die EXEC-Anweisung wird benutzt, um festzulegen, welches Programm aufzurufen ist, bzw. welche JCL-Prozedur an Stelle der Anweisung eingefügt werden soll.

Stepname

Ist nicht zwingend erforderlich, außer es wird auf den Step Bezug genommen, sollte aber angegeben werden. Anderenfalls ist keine Referenzangabe auf Stepebene möglich.

Stellungsparameter

Als Stellungsparameter kann der Programm- oder Prozeduraufruf angegeben werden.

Programmaufruf:

```
//name EXEC PGM=programmname
```

programmname muss Member einer gültigen Lademodul-Bibliothek sein. Liegt das Programm in keiner standardgemäß zugeordneten Lademodulbibliothek, muss die Ladedatei durch eine JOBLIB- oder STEPLIB-DD-Anweisung zugeordnet werden.

Prozeduraufruf:

```
//name EXEC PROC=prozedurname
```

prozedurname ist Member einer JCL-Prozedurbibliothek, oder der in der PROC-Anweisung angegebene Name einer Instream-Prozedur.

Schlüsselwort-Parameter

Schlüsselwort	Bedeutung
ACCT ADDRSPC COND REGION PERFORM TIME	Siehe Job-Anweisung
DYNAMBR	Maximale Anzahl der in einem Programm dynamisch zuweisbaren Dateien (nicht über DD-Anweisung)

Die DD-Anweisung (Daten-Definition)

```
//ddname DD operanden
```

Sie beschreibt eine Datei in einem Job und definiert die Ein-/Ausgabeeinheit für diese Datei. Außerdem wird die Verbindung zum symbolischen Namen der Datei im Programm hergestellt. Maximal können 1635 DD-Anweisungen (Größe der Task Input Table 32KB) oder 3273 (TIOT = 64KB) in einem Job liegen.

Um eine DD-Anweisung in einer Prozedur zu verändern, müssen folgende Punkte berücksichtigt werden:

- Vor den DD-Namen muss der Stepname geschrieben werden. Beispiel (//STEP1.EING1 DD ...).
- Es müssen nur die Operanden angegeben werden, die verändert werden sollen.
- Wenn mehr als eine DD-Anweisung geändert werden soll, muss die Reihenfolge mit der Reihenfolge in der Prozedur übereinstimmen.
- Wird ein DD-Name verwendet, der in der Prozedur nicht vorkommt, wird die Anweisung hinzugefügt.

DD-Name

Benennt die Anweisung. DD-Namen in einem Step müssen eindeutig sein.

Wird ein DD-Name weggelassen, wird die Datei mit dem zuletzt angegebenen DD-Namen verkettet.

Folgende DD-Namen dürfen nicht beliebig benutzt werden

DD-Namen	Bedeutung
JOBLIB	Private Bibliothek für PGM=
STEPLIB	
SYSMDUMP	Dump-Ausgaben
SYSUDUMP	
SYSABEND	
JOBCAT	Sonstige
SYSCKEOV	
SYSCHK	
IMAGELIB	
JESJCL	Reserviert für JES
JESJCLIN	
JESMSGLG	
JESYSMSG	

Stellungsparameter

Nachfolgende Parameter können nur alternativ angegeben werden.

Stellungsparameter	Bedeutung
*	Der Beginn von Instream-Daten wird definiert, die unmittelbar dahinter folgen. Das Ende der Daten kann durch eine Anweisung /* gekennzeichnet werden.

DATA, DLM=xx	DATA wird benutzt, wenn Daten auf den Stellen 1 und 2 // aufweisen. Dies ist der Fall, wenn JCL-Statements als Instream-Daten verarbeitet werden. Das Ende der Daten wird durch die Anweisung /* gekennzeichnet.
	DLM= definiert ein Begrenzungszeichen (2 Byte lang).

DUMMY	Schaltet eine Datei aus. Die übrigen Operanden können im Anschluss an DUMMY geschrieben werden. Sie werden auf syntaktische Fehler geprüft, ansonsten aber ignoriert. DSN=NULLFILE ist gleichbedeutend mit DUMMY. Wenn im DCB des Programms keine BLKSIZE beschrieben ist, muss die Angabe in der DD-Anweisung erfolgen, da sonst ein Fehler auftritt.

Beispiel

```
//AUSGABE DD DUMMY,DCB=BLKSIZE=132
```

Schlüsselwortparameter

Wie bereits erwähnt, können Schlüsselworte in beliebiger Reihenfolge kodiert werden, da sie aufgrund ihrer Namen erkannt werden. Wir wollen hier nicht alle exotischen Operanden anführen, sondern nur diejenigen, welche im Alltagsleben am häufigsten gebraucht werden (der Schwerpunkt des Buches ist REXX). Die Details zu den jeweiligen Operanden können im Kapitel 8 nachgelesen werden.

Schlüsselwort	Bedeutung
DSN	Dateiname für die Verarbeitung
DISP	Definiert, ob die Datei vorhanden ist, oder nicht, die Art des Zugriffes (exklusiv oder nicht) und was bei Ende der Verarbeitung im Normalfall (normal END – 2. Parameter) sowie im Falle eines Abbruchs (ABEND – 3. Parameter) passieren soll.)
UNIT	Gerätegruppenangabe, im Unternehmen nachzufragen. Allgemein gültige Angaben wie DASD (Direct Access Storage Device – Platte, sind nicht erwünscht.
SPACE	Platz-Größenangaben für neu zu erstellende Dateien auf Platte
DCB	Data Control Block, beschreibt den physischen Aufbau einer Datei. Meist nur nötig bei Neuanlage.
LABEL	Nur bei Bandverarbeitung. Liefert meist Angaben für Kennsätze, die bei der Verarbeitung von Kundendatenträgern benötigt werden, oder definiert Sperrfristen (Verfalldatum)

Das Zusammenwirken

Die eigentliche Schwierigkeit bei der Erstellung von Job Control Abläufen liegt nicht an der JCL an sich, sondern in einem detaillierten Wissen um die Arbeitsweise der Programme, für welche die JCL erstellt werden soll.

Um dem gerecht zu werden müsste im Grunde jede Open-Instruktion, jedes Select-Statement in einem (Cobol)Programm gekannt werden und auch, welche Verarbeitung das Programm unter dem jeweiligen Synonym durchführt. Nur dann kann sinnvoll entschieden werden, wie die dazu passende Job Control auszusehen hat.

Wir wollen hier anhand einiger Beispiele zeigen, wie die Parameter der JCL zusammenspielen. Dazu benutzen wir einige der gängigen Utilities (IEBGENER und DFSORT).