ISSC Shanghai, AMS, GCG

MVS JCL and Utilities
For Entry Level Training

ISSC SH
Yinjun JIA
Version 1.1

2004/03

Course Description

- **Purpose**

  This Classroom course of 3 days is to teach you how to use MVS job control language (JCL) and selected MVS utility program in an online batch environment.

- **Audience**

  This course is intended for personnel who want to use MVS JCL and MVS utilities.

- **Prerequisites**

  Before taking this course, you should study MVS concepts and facilities or have equivalent knowledge.

- **Objectives**

  After completing this course, you should be able to :

  – Code basic JCL statements using proper syntax and coding rules
  – Identify storage management subsystem requirements
  – Code in-stream and cataloged procedures
  – Use symbolic parameters in procedures
  – Code procedure overrides and modifications
  – Use selected utility programs

Agenda

	Day 1	Day 2	Day 3
1st	**Data Organization** **Introduction to JCL** **JOB,EXEC, and DD Statements** **EXERCISE 1**	**Introduce to Utilities** **EXERCISE 4**	**Advance Topics** **EXERCISE 6**
2nd	**JOB,EXEC, and DD Statements (Cont.)** **EXERCISE 2** **DD Parameters** **EXERCISE 3**	**Introduce to AMS Utilities** **Procedures** **EXERCISE 5**	**AMS JCL Standards** **EXERCISE 6(cont.)**

Table of contents

Data Organization

Overview

What This Unit About:

Basic data management terms and data set organizations are defined in this topic.

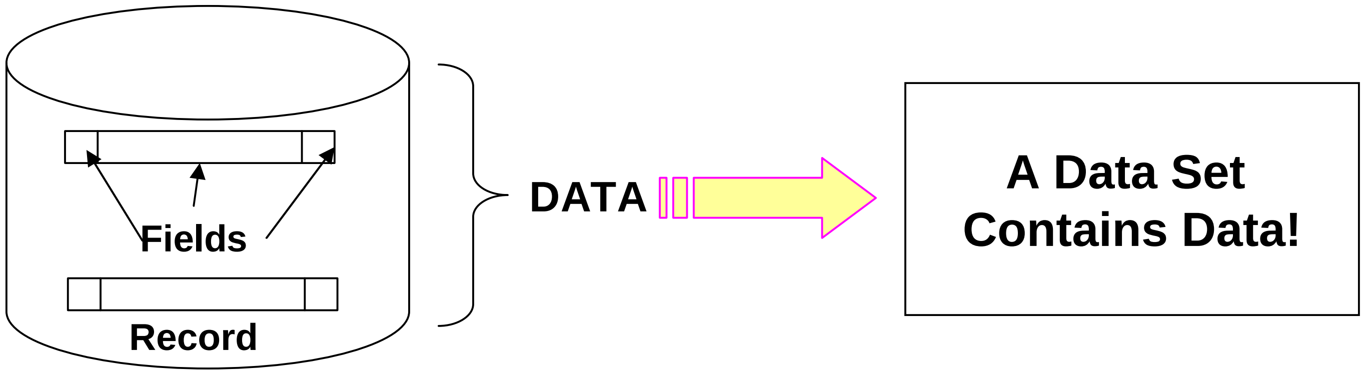
Topic Objectives:

Successful completion of topics in this unit should enable you to:

•Define basic data management terms
•Describe the differences between a sequential data set and partitional data set (PDS)
•Define Virtual Storage Access Method (VSAM) data sets
•Identity the differences between the DSNNAME and SPACE parameters for a sequential data set and partitional data set
•Describe the purpose of the Volume Table of Contents (VTOC) and the term Data Set Control Block (DSCB)

Data Organization

Data Management Terms



DATA: Information provideed to the computer for processing

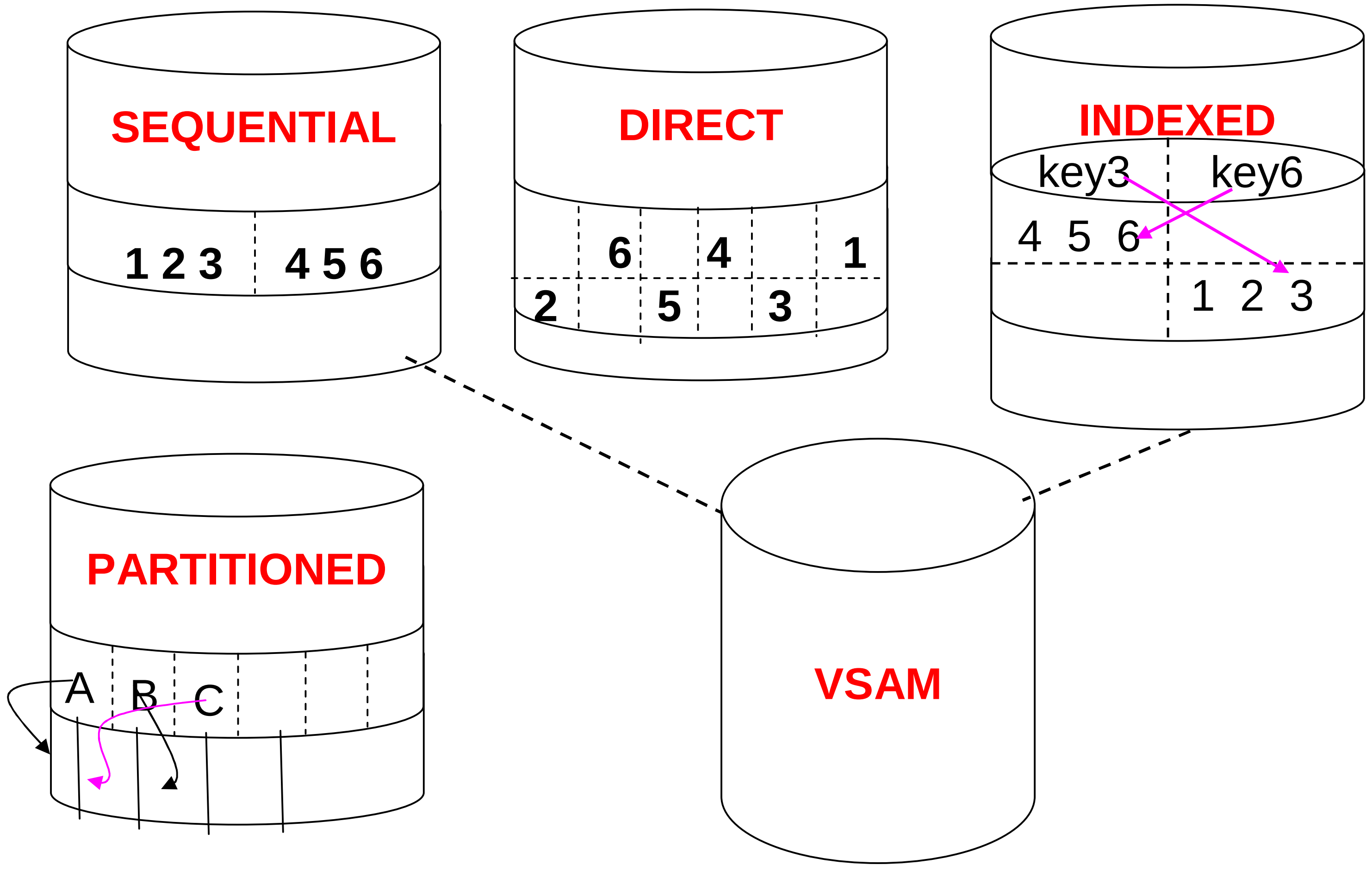
FIELD: A specified area used for a particular category of data

RECORD: A collection of related data

DATA SET: A file of related records.

Data Organization

Data Set Organizaitons



Data Organization

Data Set Organizaitons(Cont.)

VSAM:

Virtual Storage Access Method(VASM) have serveral different internal organizations. Depending on the internal organization, VSAM can mimic a sequential, direct or indexed sequential. VSAM data set can reside on DASD only.

Direct:

Records are written into the data set in an order determined by the program. Direct data sets can reside on DASD only.

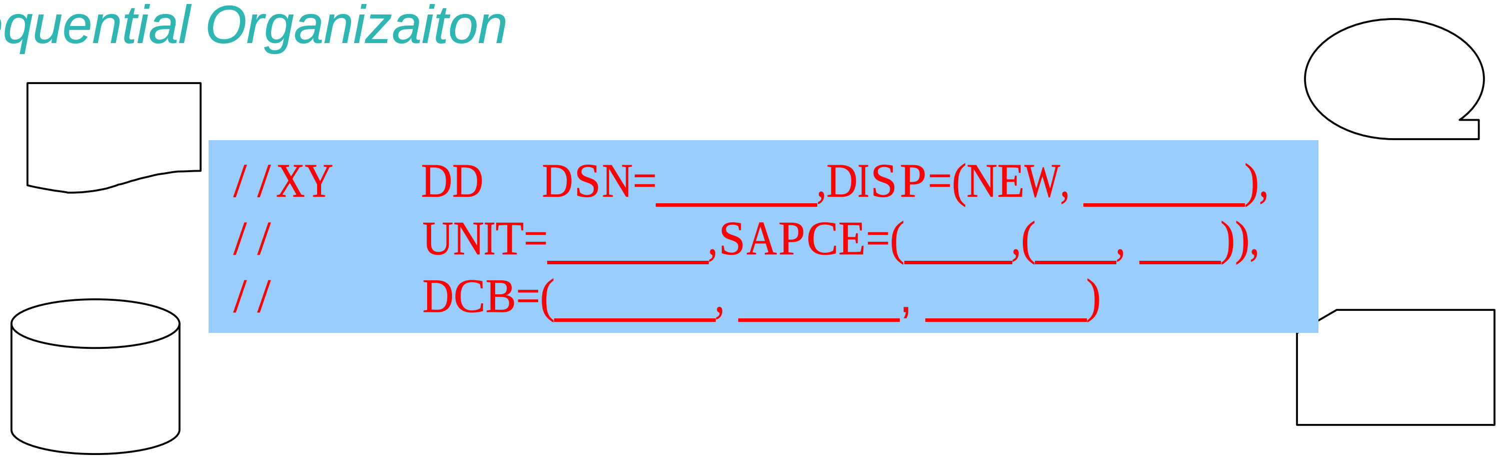
Indexed Sequential:

When first loaded, records are in sequential. Various index records are built and maintained by the organization automatically. This organization is not recommanded and is not supported in MVS new versions.

DASD **à** Directly Access Storage Device

Data Organization

Sequential Organizaiton



```
//XY     DD    DSN=________,DISP=(NEW, ________),
//             UNIT=________,SAPCE=(_____,(____, ____)),
//             DCB=(________, ________, ________)
```

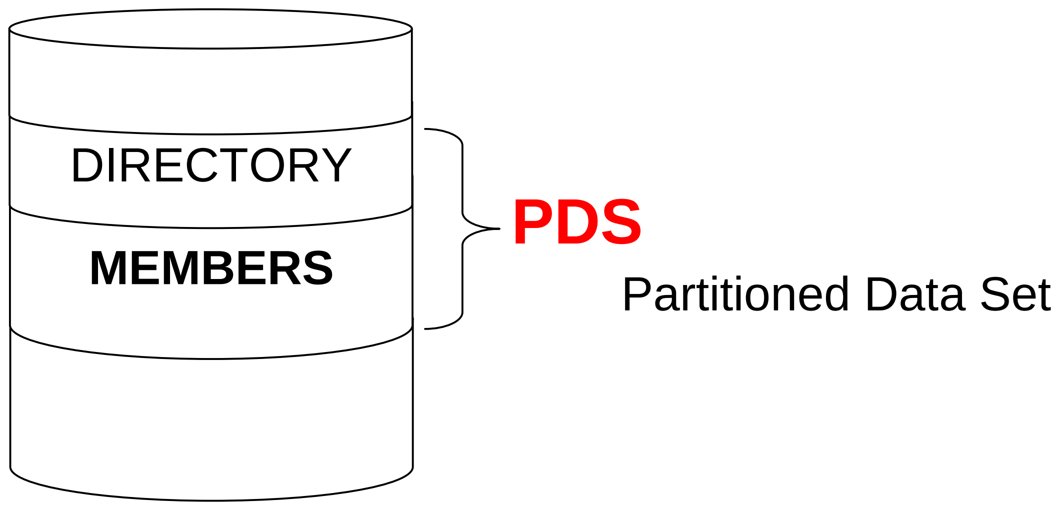
- Records are written in time of arrival order.
- Records can be processed starting at the beginning or end of the data set.
- Records can be added to the end of the data set, but can not be inserted between two existing records.
- All device types supported.
- Multiple volumes allowed

Example:

```
//AB     DD    DSN=ABC,DISP=(NEW,CATLG,DELETE),
//             UNIT=SYSDA,SAPCE=(TRK,(20,10)),
//             DCB=(LRECL=80,RECFM=FB)
```

Data Organization

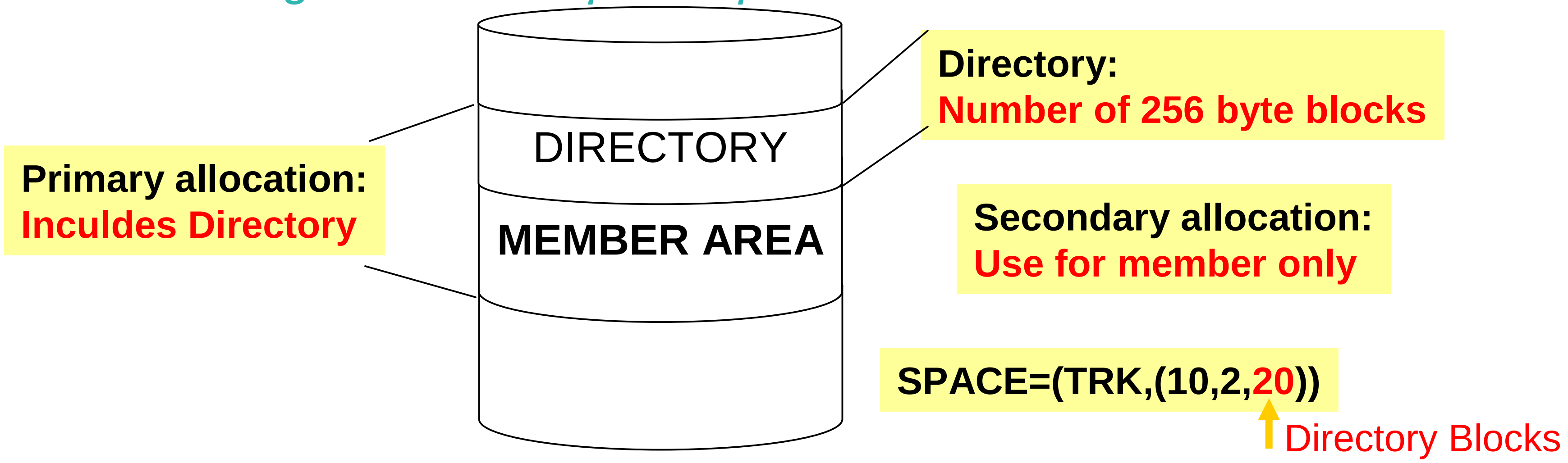
Partitioned Organizaiton



- A group of records called a member is written in sequential order and assigned a Name in the directory.
- A DIRECTORY is an index that is used by the control program to locate a member in a PDS.
- The directory is kept in ascending sequence
- Individual members can be written in any order.
- A partitioned data set is commonly called a library.
- A PDS can reside on a single volume.

Data Organization

Partitioned Organizaiton – Space Specification



- The DIRECTORY is made up of 256 byte blocks taken from the primary allocation
- Secondary allocation is not used for the directory.
- It is possible t run out of space in a PDS in two ways:
  - There can be no space left for members. This implies that no space is left on the volume if secondary allocation was requested
  - There can be no space left in the directory.

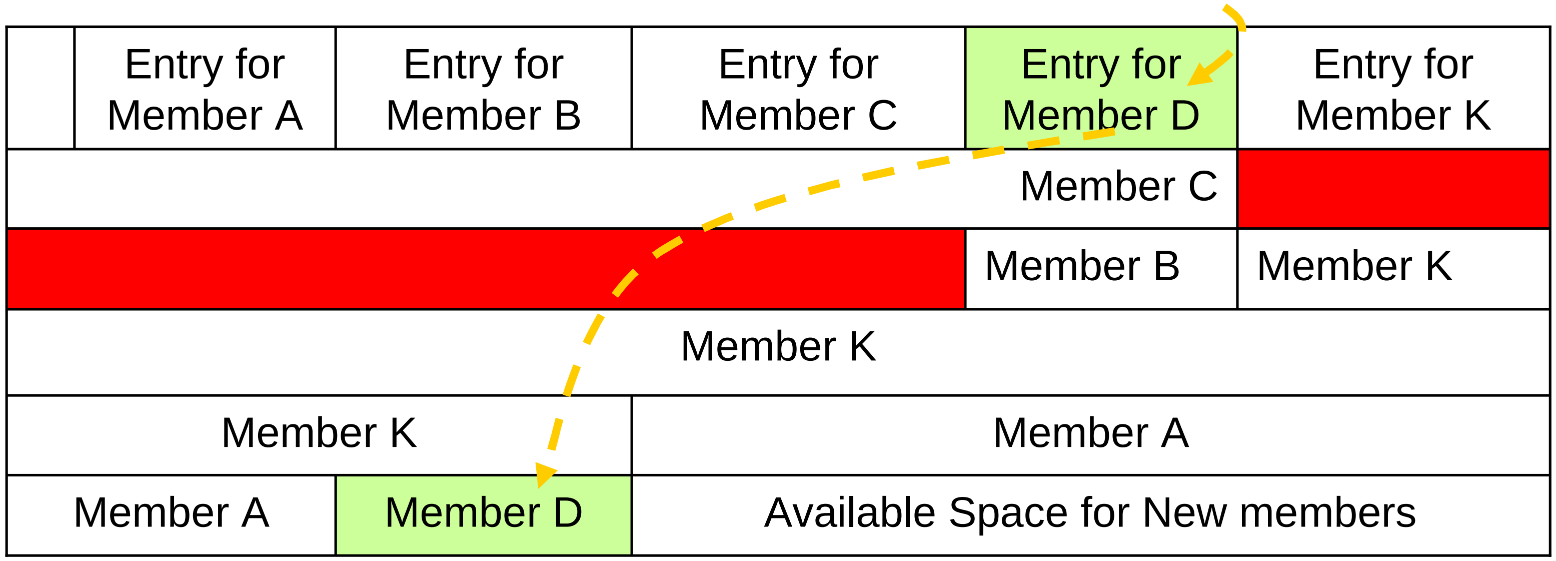
Example:

```
//AB      DD    DSN=ABC,DISP=(NEW,CATLG,DELETE),
//              UNIT=SYSDA,SAPCE=(TRK,(20,10,7)),
//              DCB=(LRECL=80,RECFM=FB)
```

Data Organization

Partitioned Organizaiton - Addition

```
//OUTPUT   DD  DSN=MY.PDS(D),DISP=OLD
```



- The new member named D have been added to the end of the existing PDS named MY.PDS even though there appears to be enough room for member D following member C
- New members and edited versions of existing members are written at the end of the data set.
- Space for deleted members can be reclaimed for use by compressing the PDS.

Data Organization

PDSE

What's new? PDSE

???

Data Organization
PDSE (cont.)

- Externally, a PDSE is similar to a partitioned data set. A PDSE has a different internal format than a PDS which gives them additional usability.
- A PDSE can only be created while SMS is active and must reside on an SMS managed volume.
- Functional benefits of PDSEs:
  - ü Eliminates the need for data set compression
  - ü Allows the directory to expand after allocation
  - ü Ensures data integrity at member level within a processor and across processors with corss system coupling facility(XCF)

Data Organization

VSAM Organizations

- KSDS – KEY SEQUENCED DATA SET
- RRDS – RELATIVE RECORD DATA SET
- ESDS – ENTRY SEQUENCE DATA SET
- LDS - LINEAR DATA SET

- **KSDS:** Key-sequenced data set contains records in ascending collating sequence, and can be accessed by a field, called a key, or by a relative byte address.
- **RRDS:** Relative record data set contains records in order by relative record number, and can be only accessed by this number
- **ESDS:** Entry-sequenced data set contains records in the order in which they were entered. Records are added to the end of the data set, and can be accessed sequentially or randomly.
- **LDS:** Linear data set contains data that has no record boundaries. Linear data set contains no control information

- All VSAM data sets must be **cataloged**
- The program **IDCAMS** is used to provide most utility services for VSAM data sets.

Data Organization

DASD VOLUME TABLE OF CONTENTS (VTOC)

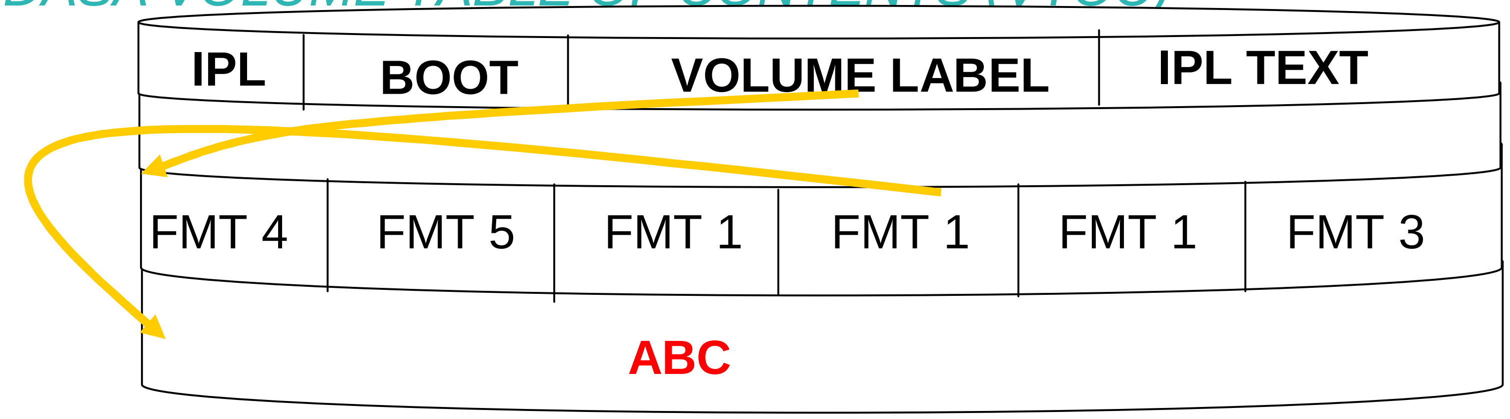
Key
44 bytes
Data Set Name

Data
96 bytes
Creation Date Expiration Date Date Last referenced Organization Record format Block size Logical record size Key length Secondary quanlity Last track used Up to 3 extents Pointer to FMT 2 or FMT 3

- Key: A 44 byte key containing the data set name
- Data: a 96 byte data area follows the key on the track.

Data Organization

DASA VOLUME TABLE OF CONTENTS (VTOC)



- All DASD volumes have a standard volume label that contains the volume serial number and pointer to the VTOC
- The VTOC is a physical sequential file with different record formats
- Data Set Control Blocks (DSCBs) are the records that make up a VTOC
- There are seven DSCB formats:
  - üFormat 0: This represents an available DSCB in the VTOC
  - üFormat 1: This decribes an existing DASD data set
  - üFormat 2: ISAM uses this to describe ISAM index areas and options
  - üFormat 3: This decribes additional (up to 13) data set extents
  - üFormat 4: This decribes the VTOC. It appears first in the VTOC.
  - üFormat 5: This decribes available space on the volume
  - üFormat 6: This decribes split cylinder allocations. (This is not used by MVS)

Table of contents

Introduction to JCL

JCLって何？

JCLって何？

＝　Job　Control　Language

ジョブ　制御　　　言語

‘言語’　というからには、

プログラミング言語なの？

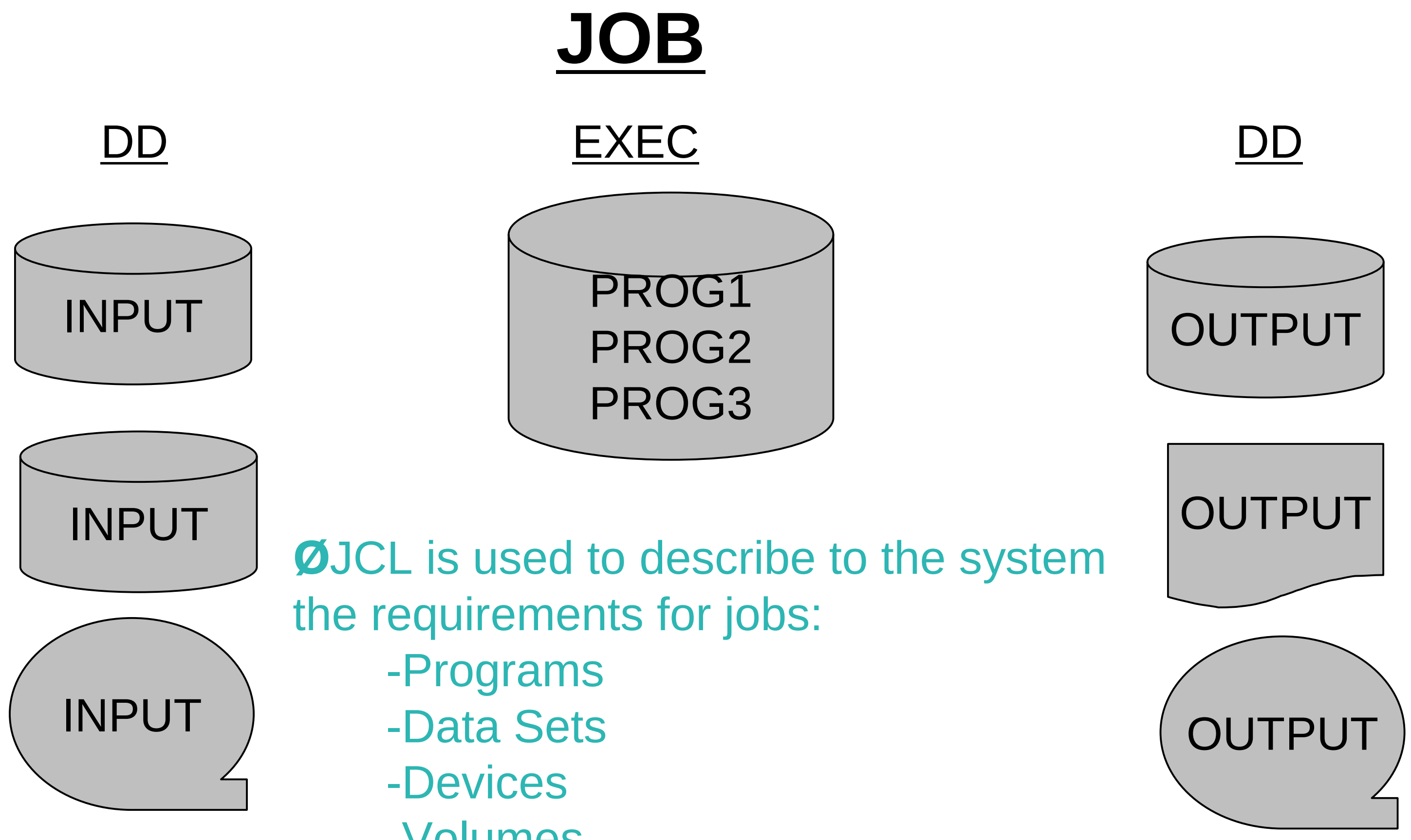
Yes !!

ただし、プログラムを実行する

プログラミング言語である。

Introduction to JCL

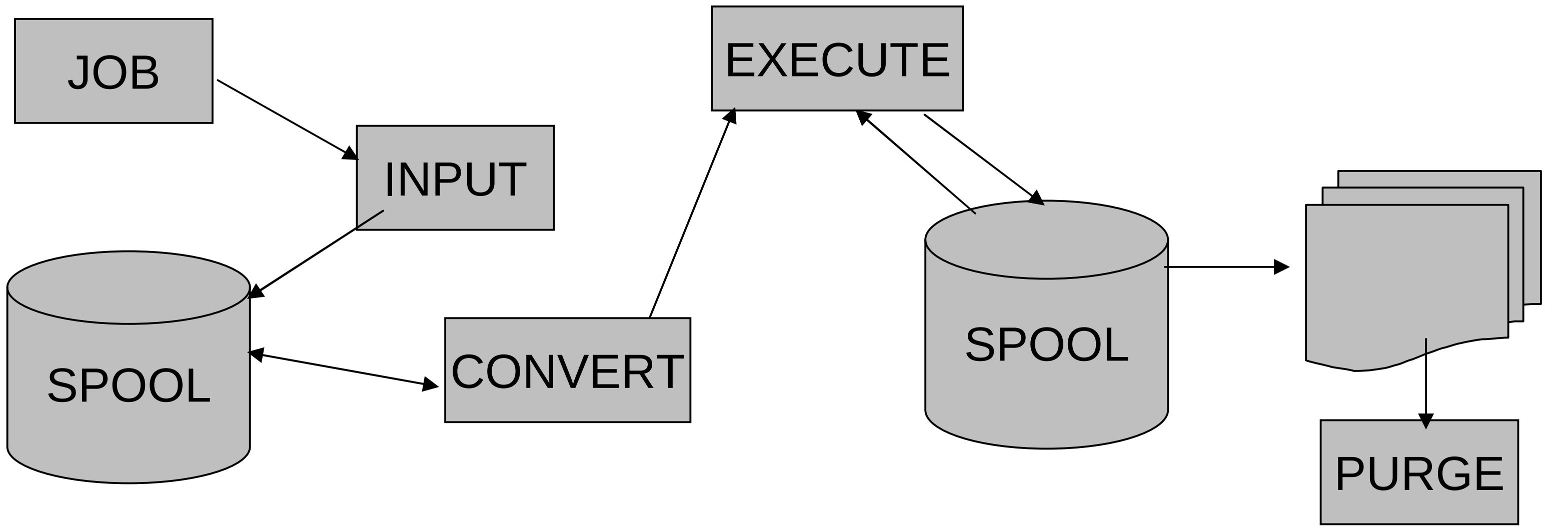
JOB CONTROL LANGUAGE



- JCL is used to describe to the system the requirements for jobs:
  - -Programs
  - -Data Sets
  - -Devices
  - -Volumes
  - -Priorities
- JCL statements are used to input this information to the system

Introduction to JCL

JES RESPONSIBILITIES



ØMVS shares the management of work with a job entry subsystem(JES), which manages work before and after execution.

ØThe general function of JES:

-JES accepts work

-Prepares work for execution

-Temporarily stores work on DASD until MVS is ready to accept it

-Selects jobs for MVS execution

-Handles printed output

-When the work completes, JES purges it from the system

Introduction to JCL

JCL Statements

- Ø JOB
- Ø EXEC
- Ø DD
- Ø PROC
- Ø PEND
- Ø COMMENT
- Ø NULL
- Ø DELIMITER
- Ø OUTPUT
- Ø JCL COMMAND*
- Ø CNTL
- Ø ENDCNTL
- Ø COMMAND
- Ø IF/THEN/ELSE/ENDIF
- Ø INCLUDE
- Ø JCLLIB
- Ø SET
- Ø XMIT**

Introduction to JCL

JOB Statement

Ø Defines a JOB & JOB related information to the system

o Accounting Information
o Programmer
o Class
o Storage Required
o Conditional testing

Notes:

ØThe JOB statement must be the first JCL statement in each JOB
ØThe JOB statement marks the beginning of a job

Introduction to JCL

EXECUTE Statements

- Ø Defines JOB step & JOB step related information to the system
  - o What program (or procedure) to run
  - o Conditional testing
  - o Parameters to be passed to program

Notes:

ØThe EXECUTE statement marks the beginning of a step in the job
ØThe EXECUTE statement must be the first JCL statement in each job
ØA job can have a maximum of 255 job steps

Introduction to JCL

DATA definition Statement

- Ø Defines data requirements for the program
  - o Describe a DATASET
  - o Specify input and output resources for a DATASET

Notes:

ØDD statements can appear in any order following the EXEC statement

ØDD statements are used to route print/punch data sets to JES for processing

Introduction to JCL

JCL Sample

```
//EV6098CP JOB (F9500B,SA00X,31),EV6098,
//        CLASS=M,MSGCLASS=R,REGION=4500K,NOTIFY=EV6098
//*-------------------------------------------------------------------*
//COPY1  PROC SDSN=,DDSN=
//STEP1  EXEC PGM=IEBCOPY
//SYSPRINT DD SYSOUT=*
//SYSUT1   DD DSN=&SDSN,DISP=SHR
//SYSUT2   DD DSN=&DDSN,DISP=OLD
//SYSUT3   DD UNIT=SYSDA,SPACE=(CYL,(1,1))
//SYSUT4   DD UNIT=SYSDA,SPACE=(CYL,(1,1))
//       PEND
//*===================================================================*
//STEP1  EXEC COPY1,
//          SDSN='SP30.DS.JCLM',
//          DDSN='WD01I.SP30.DS.JCLM.UPMAXC'
//SYSIN      DD *
   COPY  OUTDD=SYSUT2
         INDD=((SYSUT1,R))
//           DD DSN=WD01I.SP30.DS.JCLM.JCLLIST,DISP=SHR
/*
```

Introduction to JCL

JCL errors

- If a JCL syntax error is detected during conversion, the entire job is bypassed
- If a JCL error is detected, or inability to allocate space is detected during Scheduling of a step, the remaining steps of the job are bypassed

- If an executing program is abnormally terminated(ABEND), all remaining job steps in the job are bypassed, Messages indicate this condition.

- Programs that do not ABEND assign a return code to signify a certain condition
- Most IBM-developed programs produce standard return code: 0,4,8,12,16

Table of contents

JOB, EXEC, and DD Statements

JCL Statement Format

```
//NAME   OPERATION  PARAMETER,
// PARAMATER,PARAMETER
```

- ØA JCL statement consisits of one or more than 80 byte records
- ØA continued JCL statement can begin in col 4-16
- ØEach statement is divided into the following five fields:

IDENTIFIER field

Indicated a JCL statement

-// in columns 1 and 2 of all JCL statements except the delimiter

-/* or installation designated character in columns 1 and 2 as a delimiter

-//* in columns 1,2 and 3 depicts a comment statement

NAME field

Identifies a statement so that it can be referred to later

-Must begin in column 3

-1-8 charactors in length(alphanumeric or national(#,@,$))

-First character must be alphabetic or national

JOB, EXEC, and DD Statements

JCL Statement Format – cons.

OPERATION field

Specifies the type of statement or command
-Follows the NAME field
-Must be preceded and followed by at least one blank

PARAMETER field

Contains paramters separated by commas
-Follows the OPERATION field
-Must be preceded and followed by at least one blank
-Consists of two type: Positional and Keyword

COMMENT field

Can contain any information
-Follows the PARAMETER field
-Must be preceded by at least one blank
-Difficult to use, Comment Statement is recommended

JOB, EXEC, and DD Statements

Parameters

```
//NAME   OPERATION  P1,P2,P3,K1=A,
// K2=B,K3=(P1,K1=T)
```

A DD statement has two kinds of parameters: Positional and Keyword. All parameters are optional.

-**Positional:** if needed, must be coded first in the order described in the JCL manual.

-If a leading positional parameter is omitted and trailing positional parameters follow,code a comma to indicate its absence

-**Keyword:** code keyword parameters,in any order, after required positional parameters.

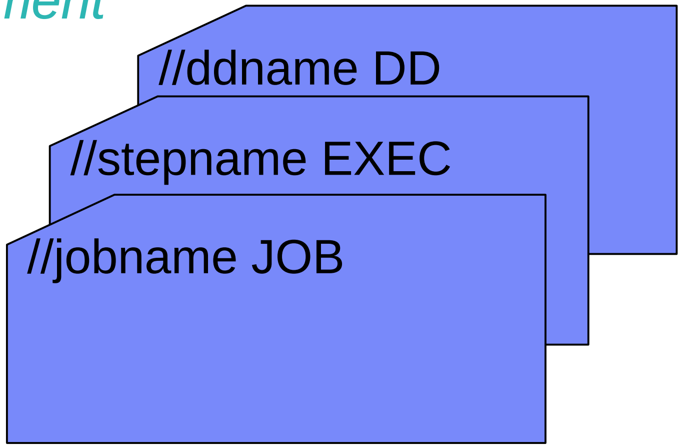
-Keyword parameters are indicated by coding the keyword following by an “=” sign and the keyword value

-Commas are not coded to indicate the omission of keyword parameters

Subparameter

JOB, EXEC, and DD Statements

The JOB Statement



The JOB statement must be:

- ØThe first statement in the job
- ØThe only JOB statement in the job

The JOBNAME: (required)

ØShould be unique

- Ø1-8 charactors in length(alphanumeric or national(#,@,$))
- ØFirst character must be alphabetic or national
- ØMust be followed by at least one blank

JOB, EXEC, and DD Statements

JOB Statement syntax

//jobname JOB positionals,keywords

// in column 1 and 2

JOBNAME is required

JOB in the operation field

Positional parameters

- ØACCOUNTING INFORMATION(maximun of 142 chars)
- ØPROGRAMMER NAME(20 Chars)

The following keyword parameters, a subset of those available, will be covered in this course:

ØCLASS	assigns the job to a jobclass
ØCOND	specifies rules for conditional job execution
ØLINES	limits print output before system takes action
ØMSGCLASS	assigns the job log to an output class
ØMSGLEVEL	shows information t be placed in the job log
ØNOTIFY	notifies the TSO user when a job is complete
ØRD	controls the use of checkpoint/restart
ØREGION	specifies maximum virtual storage size for a job
ØRESTART	controls the restart of a job
ØTIME	specifies the maximum time a job can execute
ØTYPRUN	requests special job processing

JOB, EXEC, and DD Statements

CLASS & MSGCLASS parameter

```
//jobname  JOB  positionals,CLASS=jobclass,MSGCLASS=class
```

CLASS:

ØThe class places your job into a JES input queue class
ØCLASS is one character, A-Z or 0-9
ØIf you donot specify a class, JES uses the installation default specified

MSGCLASS:

ØMSGCLASS controls the destination of the job log
ØCLASS is one character, A-Z or 0-9
ØIf you donot specify a class, JES uses the installation default specified

JOB, EXEC, and DD Statements

MSGLEVEL parameter

//jobname JOB positionals, MSGLEVEL=(x,y)

The **first** subparameter of MSGLEVEL controls which statements will be printed in the job log

- 0 Print only the JOB statement
- 1 Print all JCL and JES statement including all statements in precedures
- 2 Print only submitted JCL and JES statements. Statements in proc are not printed

The **second** subparameter of MSGLEVEL controls which messages will be printed in the job log

- 0 If normal, print only JCL messages, otherwise, print all messages
- 1 All messages are printed regardless of how the job terminates

JOB, EXEC, and DD Statements

NOTIFY & TYPRUN parameter

//jobname JOB positionals, NOTIFY=tsoid,TYPRUN=option

The **NOTIFY** parameter causes the system to notify the TSO user specified on the NOTIFY parameter when the job completes.

The **TYPRUN** parameter modifies the way JES process your job
ØSCAN The JCL is scanned for syntax errors but is not executed
ØHOLD The job is held in the input queue. The operator must release the job to execute it
ØCOPY The JCL is copied as submitted to the SYSOUT class specified in the MSGCLASS parameter. The job is scanned for syntax errors but is not executed

JOB, EXEC, and DD Statements

REGION parameter

```
//jobname  JOB  positionals, REGION=100M
//stepname EXEC PGM=ABC, REGION=100K
```

The **REGION** parameter requests space needed for the JOB. It can be coded on the JOB or EXEC statements.

Ø When REGION is specified on the JOB statement, it applies to each step in the job and overrides the REGION parameter on each EXEC statement

Ø A job is ABENDED if some step needs a larger REGION size or if the REGION value can not be obtained

Ø When REGION is specified on the EXEC statement, it specifies the amount of space required by the step. It should be used when different steps require different amounts of space

Ø REGION can be specified with the value equals to K(kilobytes) or M(megabytes)

JOB, EXEC, and DD Statements

LINE parameter

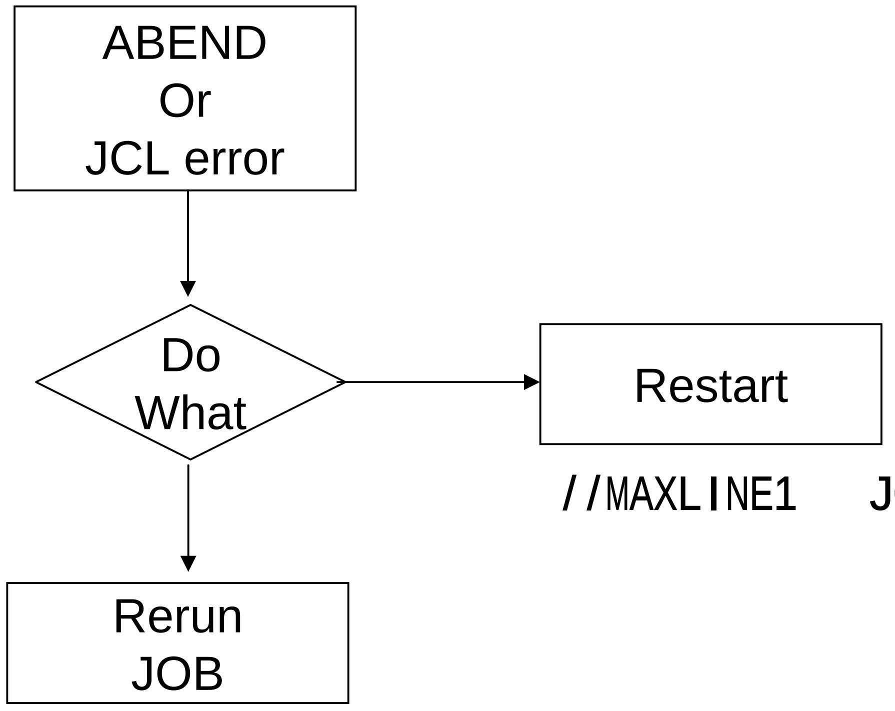
//jobname JOB positionals, LINE=(nnnnnn,action)

The **LINE** parameter is used to limit the amount of output to be printed for a job's SYSOUT datasets. The system can be take any of the following actions if the maximum is exceeded

- CANCEL cancel job
- DUMP dump storage
- WARNING send warning message to operator

```
//MAXLINE1   JOB  LINES=(50,CANCEL)
//MAXLINE2   JOB  LINES=(50,DUMP)
//MAXLINE2   JOB  LINES=(50,WARNING)
```

JOB, EXEC, and DD Statements
RESTART parameter



```
//MAXLINE1   JOB   ___,___,RESTART=STEP10
```

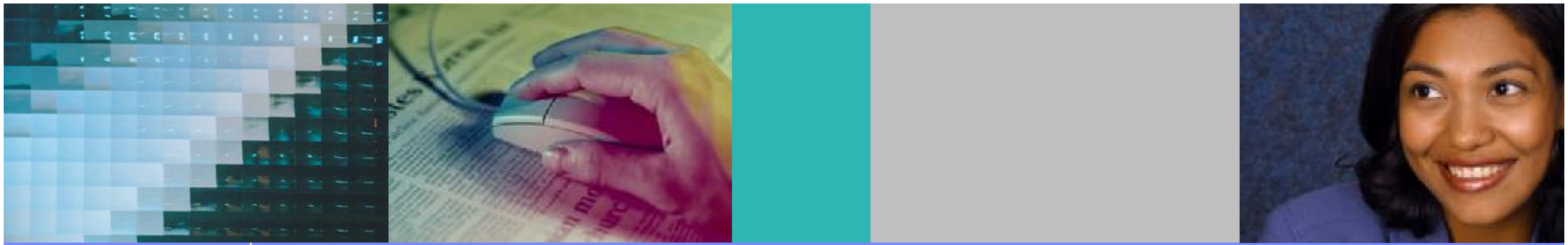
To use the RESTART option, stepnames must be unique

JOB, EXEC, and DD Statements

JOB statement - examples

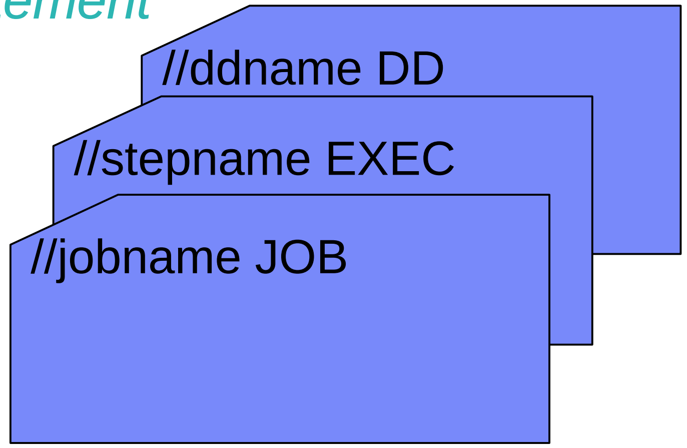
```
//TEST1   JOB  (DEPT378,399216),LEON,CLASS=T

//TEST2   JOB  DEPT378,TOM,CLASS=M,MSGLEVEL=(1,1)

//TEST3   JOB

//SYSTEM  JOB ,SYSTEM,CLASS=S,MSGLEVEL=(0,0)

//SUBMIT  JOB MACHE999999,’R. J. Y’,NOTIFY=TSOID9,CLASS=A,REGION=512K
```

EXERCISE 1



JOB, EXEC, and DD Statements

The EXEC Statement



The EXEC statement defines the beginning of a step in a job or a proc

An EXEC statement is required for each job step

Maximum of 255 steps(EXEC statements) in a single job

The STEPNAME:

- 1-8 charactors in length(alphanumeric or national(#,@,$))
- First character must be alphabetic or national
- Must be followed by at least one blank

IT IS A GOOD PRACTICE TO USE UNIQUE STEPNAMES WITH IN A JOB

JOB, EXEC, and DD Statements

EXEC Statement syntax

//stepname EXEC positionals,keywords

// in column 1 and 2

STEPNAME is not required,but is recommended

Operation of EXEC

Positional parameters

- ØPGM= or PROC=

The following lists a subset of keyword parameters:

ØACCT	Allows job steps to be charged to different account codes
ØADDRSPC	Select a type of storage require for this step
ØCOND	Specifies rules for conditional step execution
ØDPRTY	Request a dispatch priority
ØPARM	Passes information to the program
ØREGION	Specifies the virtual storage size for a step
ØTIME	Specifies the maximum time the step is allowed to execute

JOB, EXEC, and DD Statements

PROGRAM EXECUTION & TIME PARAMTERS

Program execution:

By default, SYS1.LINKLIB is searched for the program t be executed

Private user libraries can be searched before SYS1.LINKLIB by using:

- ØA JOBLIB DD statement
- ØA STEPLIB DD statement

TIME=(minutes,seconds)

The TIME parameter can be used to specify the maximum length of time that a job or job step is to use the processor

- ØIf coded on the JOB statement, this is the total time for all steps
- ØIf coded on the EXEC statement , it is maximum time for this step

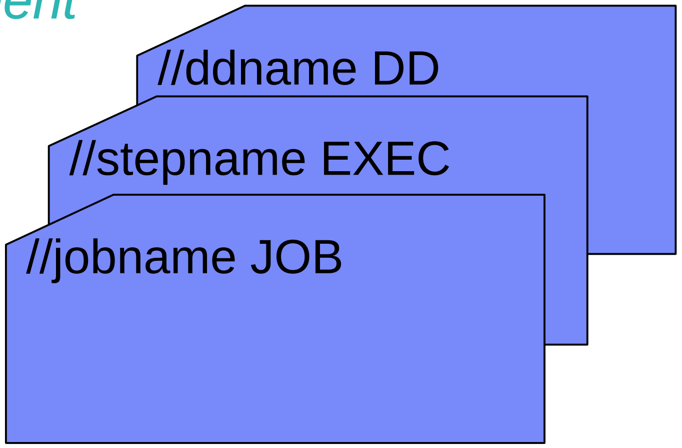
JOB, EXEC, and DD Statements

EXEC STATEMENT - SAMPLE

```
//STEP1  EXEC PGM=FIRST

//STEP2  EXEC PGM=SECOND,PARM=94.33,TIME=(,10)

//STEP3  EXEC PGM=THIRD,PARM=’94/12/31’

//STEP4  EXEC PGM=FOURTH,PARM=‘LSIT,MAP,XREF’
```

JOB, EXEC, and DD Statements

The DD Statement



ØThe DD statement is used to describe a dataset and specify the input and output resources needed for the dataset

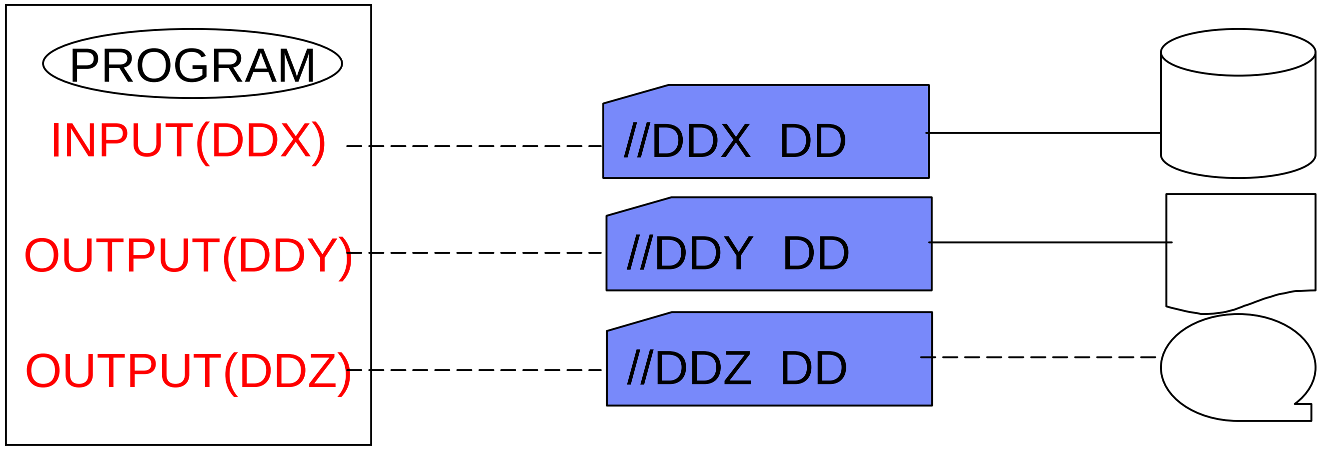
ØA DD statement is normally required for each dataset that is to be processed in a step

ØAll DD statements for a single step must follow the EXEC statement

ØDD statements for a single step can usually be in any order

JOB, EXEC, and DD Statements

WHY DATA DEFINITION (DD)?



ØThe DD statement is necessary because the program does not reference a data set directly. The name of DD statement is coded in the program. When a dataset is OPENED for processing,the name is used to locate the proper DD statement

ØDDNAMES following the same rules for all names in JCL

ØEach DD statement should have a unique DDNAME with in a step

ØAvoid coding DDNAMES that begin with ‘SYS’,’JOB’,’STEP’

JOB, EXEC, and DD Statements

DD STATEMENT

```
//DDX    DD DSN=A,DISP=OLD
//            DD DSN=B,DISP=OLD

//DDY   DD *
    DATADDDDDDDDDDDDDDDDDDDDDDDDDDDDD
/*

//DDZ   DD DATA
//DDDDDDDDDDDDDDDDDD
//DDDDDDDDDDDDDDDDDDDDD
/*

//DDS   DD DATA,DLM=ZZ
//EEEEEEEEEEEEE
//DDDDDDDDDDDDDDDDD
/*
ZZ

//DDXX DD SYSOUT=*
```

Table of contents

DD Parameters

DD STATEMENT SYNTAX

```
//ddname  DD positional,keywords

//XYZ   DD DSN=____,DISP=____,
//      UNIT=_____,SPACE=_____,VOL=____,
//      DCB=_____
```

ØPositional

*	Begin an in-stream data set
DATA	Begin an in-stream data set; position 1 and 2 must be '//'
DUMMY	Specifies no allocation

ØKeyword

DCB	Provide blocking and other information to the system
DDNAME	postpones data set definition
DEST	Routes a SYSOUT data set to a specified location
DISP	Specifies the status for a data set
DLM	Specifies terminating charactors for an instream data set
DSN	nams a data set
FCB	Specifies printer forms control

DD Parameters

DD STATEMENT SYNTAX

Keyword	
FREE	Specifies when to deallocate a data set
HOLD	Holds SYSOUT data for later processing
OUTLIM	Limits the number of records in a SYSOUT dataset
OUTPUT	Associates a SYSOUT data set with an OUTPUT statement
SPACE	Assigns space to DASD data set
SYSOUT	Defines a SYSOUT data set
UNIT	Requests device allocation
VOL	Identifies the volume on a device

DD Parameters

Data Set Naming

Permanent Data Set Naming

SEQUENTIAL	DSN=MY.FIRST.DATA
PDS	DSN=SX01I.JCLEDU.JCLBK(JOB11)

TEMPORAY(WORK) DATA SETS

DSN=&&SORTOUT
DSN=&&TEMP(MEM1)

BACKWARD REFERENCE

DSN=*.ddname
DSN=*.stepname.ddname

DSN=*.DD1
DSN=*.STEP2.DD5

DD Parameters
Disposition Parameter

DISP=(initial,normal,abnormal)

Initial è
NEW
OLD
SHR
MOD(existing -> append;no->new)
Normal è
DELETE,KEEP,
PASS (effective only within a job)
CATLG,UNCATLG

Abnormal è DELETE,KEEP,CATLG,UNCATLG

DD Parameters

UNIT Parameter

UNIT=(____,count,defer)

Code the UNIT parameter to ask the system to place the dataset on

- A specific device
- A certain type or group of devices
- The same device as another data set

UNIT=device number	UNIT=130
UNIT=generic unit	UNIT=3390
UNIT=esoteric name	UNIT=SYSDA

DD Parameters

VOLUME SPECIFICATION

VOL=(,RETAIN,COUNT,SER=(……),REF=)

VOL=SER=123456

VOL=(,,,10)

Code the VOL to identify the volume or volumes on which a dataset resides or will reside
The RETAIN subparameter is a request to keep a tape volume mounted for use later in the job

DD Parameters

SPACE SPECIFICATION

SPACE=(Unit,(Primary,Secondary),__)

SPACE=(Unit,(Primary,Secondary,Directory),__)

RLSE à release unused DASD space

DCB SPECIFICATION

LRECL – the record length
RECFM – the record format
BLKSIZE – the block length

DD Parameters

RECORD FORMATS

RECFM=F Fixed Length Record
All records in the dataset have the same length. However several fixed length formats are possible. If each block contains one record, the data set is said to be unblocked

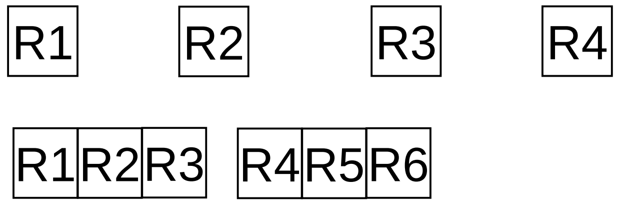
RECFM=V Variable Length Record
Not all records in the dataset have the same length.Format V allows the data set to contain variable length records and variable length blocks

RECFM=U Undefined Length Record
Format U permits the processing of records that do not conform to either F or V format

UNBLOCKED- waste a greate deal of space and should not be used

BLOCKED

Control characters:
A-ISO/ANSI control charactors(SPACE, THEN PRINT)
M-Machine control charactors(PRINT, THEN SPACE)



DD Parameters

SPECIAL DDNAMES

//JOBLIB Identifies a private library the system is to search for each pgm in the job

//STEPLIB Identifies a private library the system is to search for the pgm named in the EXEC statement PGM parameter

//SYSUDUMP Use this DD statement in a job to direct the system to produce a formatted dump of user areas

//SYSABEND Use this DD statement in a job to direct the system to produce a formatted dump of user and system areas

//SYSDUMP Use this DD statement in a job to direct the system to produce an unformatted dump of user and system areas

//SYSCHK Use this DD statement to define a checkpoint data set that the system is to write during execution of a processing program

EXERCISE 2 & 3

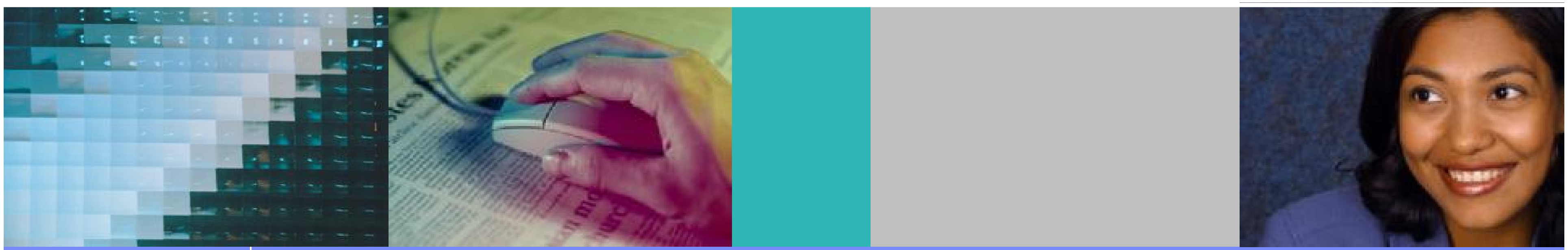


Table of contents

Introduction to Utilities

MVS Utilities

ØUtilities are IBM supplied programs which perform certain routine and frequently occuring tasks in the MVS environment

ØUtilities are used in DASD, tape, print and punch operations

ØUtilites are also used to list the contents of a VTOC, and allocate,update,delete,catalog, and uncatelog data sets

Introduction to Utilities

Classes of Utilities

ØData Set: IEB***** (data set/record oriented)
Copy,print,punch,update,reorganize,and compare data at the data set and/or record level

ØSystem: IEH***** (Volume Oriented)
List VTOC information
Copy , delete, catalog, and uncatelog datasets
Write tape labels and to add or delete data set password

ØBoth sets of utilities are controlled by JCL statements and utilitiy control statements

Introduction to Utilities

IEBGENER à CARD-TO-DISK

```
//JIAYJJ1 JOB  (ISSC#),'JIAYJ',CLASS=A,MSGCLASS=H,
//             NOTIFY=JIAYJ
//***********************************************************************
//BUILD    EXEC  PGM=IEBGENER
//SYSPRINT DD  SYSOUT=*
//SYSUT1   DD  *
  JONES   FRED     53AF    87   5701 NINE MILE ROAD
  ANDERSONDON      78AF    34   320  WESTHERIMAR, #219
//SYSUT2   DD DSN=SX01I.JCLEDU.IEBGEN1.OUT,DISP=(,CATLG,DELETE),
//         SPACE=(TRK,(1,1)),VOL=SER=DMTD02,
//         RECFM=FB,LRECL=80,UNIT=SYSDA
//SYSIN    DD  DUMMY
/*
```

ØIEBGENER is a dataset utility used to create,copy , or print sequential data sets
ØThe example allocates a dataset, and then copies in-stream data to the data set

Introduction to Utilities

IEBGENER à COPY

```
//JIAYJJ1 JOB  (ISSC#),'JIAYJ',CLASS=A,MSGCLASS=H,
//             NOTIFY=JIAYJ
//************************************************************************
//BUILD    EXEC  PGM=IEBGENER
//SYSPRINT DD  SYSOUT=*
//SYSUT1   DD  DSN=SX01I.JCLEDU.IEBGEN1.OUT,DISP=SHR
//SYSUT2   DD DSN=SX01I.JCLEDU.IEBGEN2(OUTPUT),DISP=(,CATLG,DELETE),
//         SPACE=(TRK,(1,1,20)),VOL=SER=DMTD02,
//         RECFM=FB,LRECL=80,UNIT=SYSDA
//SYSIN    DD  DUMMY
/*
```

ØThe example copy the squential input dataset to a new partitioned output dataset

Introduction to Utilities
IEBGENER à COPY/PRINT

```
//JIAYJJ3 JOB  (ISSC#),'JIAYJ',CLASS=A,MSGCLASS=H,
//             NOTIFY=JIAYJ
//*********************************************************************
//BUILD    EXEC  PGM=IEBGENER
//SYSPRINT DD  SYSOUT=*
//SYSUT1   DD  DSN=SX01I.JCLEDU.IEBGEN1.OUT,DISP=SHR
//SYSUT2   DD  SYSOUT=*
//SYSIN    DD  *
    GENERATE MAXFLDS=3
    RECORD FIELD=(10,20,,1),FIELD=(10,1,,15),FIELD=(6,5,,30)
/*
```

ØIEBGENER is used for copying or printing all or selected portions of datasets
ØThis example is to create SYSUT2 output data in a different form from the input
ØThe SYSIN DD * indicates that in-stream records,control statements follow
ØThe GENERATE statement specifies that editing is to be performed and the operand MAXFLDS=3 indicates that no more than 3 fields will be described
ØFIELD=(LENGH OF FIELD,POSITION IN INPUT,CONVERSION,POSITION IN OUTOUT)

Introduction to Utilities

IEBCOPY à COPY

```
//JIAYJJ4 JOB  (ISSC#),'JIAYJ',CLASS=A,MSGCLASS=H,
//             NOTIFY=JIAYJ
//***********************************************************************
//COPY     EXEC  PGM=IEBCOPY
//SYSPRINT DD  SYSOUT=*
//IN       DD  DSN=SX01I.JCLEDU.JCL,DISP=SHR
//OUT      DD  DSN=SX01I.JCLEDU.JCLBK,DISP=OLD
//SYSIN    DD  *
    COPY   OUTDD=OUT,INDD=IN
    SELECT MEMBER=IEBGEN1
    SELECT MEMBER=(IEBGEN2,IEBGEN3)
    SELECT MEMBER=((IEBCOPY,IEBCOPYB,R))
/*
```

ØIEBCOPY is used for copy operations on members of partitioned data sets.

ØThe format of SELECT control statement is:

SELECT MEMBER=NAME OR SELECT MEMBER=(NAME1,NAME2,NAME3)

SELECT MEMBER=((NAME,NEWNAME,REPLACE))

Introduction to Utilities

IDCAMS à Delete cataloged non-vsam dataset

```
//JIAYJJ5 JOB  (ISSC#),'JIAYJ',CLASS=A,MSGCLASS=H,
//             NOTIFY=JIAYJ
//*********************************************************************
//STEP01   EXEC  PGM=IDCAMS
//SYSPRINT DD  SYSOUT=*
//SYSIN    DD  *
    DELETE-
    SX01I.JCLEDU.IEBGEN1.OUT
/*
```

ØIDCAMS can be used to delete non-vsam datasets
ØThe DELETE statement specifies the name of the dataset to be deleted
ØRules for coding control statements for IDCAMS
- ***üColumn 1 must be blank***
- ***üA dash '-' or '+' maybe used for continuation***
- ***üMultiple control statement can be coded***

ØIDCAMS requires fewer DD statements and can used for both VSAM and non-VSAM data sets.

Introduction to Utilities

IEHLIST à LISTVTOC

```
//JIAYJJ6 JOB  (ISSC#),'JIAYJ',CLASS=A,MSGCLASS=H,
//             NOTIFY=JIAYJ
//*********************************************************************
//LISTVTOC EXEC  PGM=IEHLIST
//SYSPRINT DD  SYSOUT=*
//DD1      DD  DISP=OLD,UNIT=3390,VOL=SER=DMTD01
//DD2      DD  DISP=OLD,UNIT=3390,VOL=SER=DMTD02
//SYSIN    DD  *
    LISTVTOC FORMAT,VOL=3390=DMTD01
    LISTVTOC FORMAT,VOL=3390=DMTD02,DSNAME=SX01I.JCLEDU.IEBGEN2
/*
```

ØIEHLIST can be used to list entris in a DASD VTOC
ØThe DD1 and DD2 DD statements allocate packs needed for the LISTVTOC operation
ØSYSIN DD statement defines the control dataset. This is where IEHLIST looks for utility control statements
ØDSNAME cannot be abbreviated as DSN on a control statement
ØThe VOL parameter format is: VOL=XXXXX=YYYYYY, where XXXXX is the value specified for the UNIT parameter on the JCL DD statement and YYYYYY is the SER subparameter value

Introduction to Utilities

IEHLIST à LISTPDS

```
//JIAYJJ7 JOB  (ISSC#),'JIAYJ',CLASS=A,MSGCLASS=H,
//             NOTIFY=JIAYJ
//*********************************************************************
//LISTPDS  EXEC  PGM=IEHLIST
//SYSPRINT DD  SYSOUT=*
//NUM1     DD  DISP=OLD,UNIT=3390,VOL=SER=DMTD01
//SYSIN    DD  *
   LISTPDS FORMAT,VOL=3390=DMTD01,DSNAME=SX01I.JCLEDU.JCLBK
/*
```

ØIEHLIST can be used to list entris in a PDS directory

Introduction to Utilities

IEBPTPCH à PRINT

```
***************************** Top of Data ******************************
//JIAYJJ8 JOB  (ISSC#),'JIAYJ',CLASS=A,MSGCLASS=H,
//             NOTIFY=JIAYJ
//**********************************************************************
//PRINT    EXEC  PGM=IEBPTPCH
//SYSPRINT DD  SYSOUT=*
//SYSUT1   DD  DSN=SX01I.JCLEDU.IEBGEN1.OUT,DISP=SHR
//SYSUT2   DD  SYSOUT=*
//SYSIN    DD  *
  PRINT TYPORG=PS,MAXFLDS=3
  TITLE ITEM=('THIS A TEST FOR JCL EDUCATION',10)
  TITLE ITEM=('----------------------------------',08)
  RECORD FIELD=(8,2,,10),FIELD=(5,10,,20),FIELD=(36,25,,31)
/*
**************************** Bottom of Data ****************************
```

ØIEBPTPCH is used to print or punch all or selected protions of dataset,editing can be done on the data.

Introduction to Utilities

SORT à control statement

ØThe format of the sort field is :
SORT FIELDS=(position of First char,length of Field,data Format,A/D)
SORT FIELDS=(2,5,CH,A)
SORT FIELDS=(2,5,CH,A,9,2,CH,A)
SORT FIELDS=(2,5, A,9,2,A),FORMAT=CH

ØPGM=SORT or PGM=ICEMAN on the EXEC statement will invoke the DFSORT.

```
//JIAYJJ9 JOB  (ISSC#),'JIAYJ',CLASS=A,MSGCLASS=H,
//             NOTIFY=JIAYJ
//***********************************************************************
//SORT     EXEC  PGM=ICEMAN
//SYSOUT   DD  SYSOUT=*
//SORTIN   DD  DISP=OLD,DSN=SX01I.JCLEDU.IEBGEN1.OUT
//SORTOUT  DD  DSN=SX01I.JCLEDU.SORT.OUT,DISP=(,CATLG),
//         SPACE=(TRK,(1,1)),VOL=SER=DMDT02,
//         RECFM=FB,LRECL=80,UNIT=SYSDA
//SYSIN    DD  *
    SORT FIELDS=(9,2,CH,A,2,5,CH,A)
/*
```

Introduction to Utilities

MERGE à control statement

ØMerge: combine sorted files into a single sorted file

```
***************************** Top of Data ******************************
//JIAYJJA JOB  (ISSC#),'JIAYJ',CLASS=A,MSGCLASS=H,
//             NOTIFY=JIAYJ
//*********************************************************************
//SORT     EXEC  PGM=ICEMAN
//SYSOUT   DD  SYSOUT=*
//SORTIN01 DD  DISP=OLD,DSN=SX01I.JCLEDU.SORT.OUT1
//SORTIN02 DD  DISP=OLD,DSN=SX01I.JCLEDU.SORT.OUT2
//SORTOUT  DD  DSN=SX01I.JCLEDU.MERGE1.OUT,DISP=(,CATLG),
//         SPACE=(TRK,(1,1)),VOL=SER=DMDT02,
//         RECFM=FB,LRECL=80,UNIT=SYSDA
//SYSIN    DD  *
    MERGE FIELDS=(9,2,CH,A,2,5,CH,A)
/*
**************************** Bottom of Data ****************************
```

Introduction to Utilities

SORT/MERGE à considerations

ØFILSZ=E_____ specifies the estimated number of records in the input streams
ØFILSZ=______ specifies the total number of the records in the input streams
ØSKIPREC=_____ specifies how many records should be skipped at the beginning of the input data set
ØSTOPAFT=_____ specifies how many records should be read and sorted

EXAMPLE:
OPTION FILSZ=E10000,SKIPREC=3025,SOTPAFT=8000

ØOPTION can be used in both SORT and MERGE

Introduction to Utilities

The Utilities in JAPAN ENV.

- PROGM1,2,3 -delete or uncatalg the existing dataset
- RPD8DLD -open/close sequetial file
- SCRMSV1～4 --delete or uncatalg the existing dataset
- MSSSCR1,2
- JPD0VV1 (Multi Dataset Copy Utility)
- DCPVV1 (Multi Dataset Copy Utility)
- SYMCHK (SYSIN Automatic Generator)
- SYMCHK2 (MEC用 SYMCHK)

- CM91SCR (Extended MSSSCR2)
- CM91SELZ (Super power selection tool)
- CM91ISML (Automatic list output super tool)
- CM91MASC (Automatic ASCA check & write)
- CM91MAT3 (Automatic matching tool Vol. 3)
- CM91DUP1 (Duplicate check tool)
- CM91GEN2 (Super power layout conversion tool)
- CM91GENZ (Super power layout conversion tool & calculator tool)
- NEWSORT (DCPSORT)

For the detail, please refer to the documents from Japan AMS

EXERCISE 4

```
//JIAYJJA JOB  (ISSC#),'JIAYJ',CLASS=A,MSGCLASS=H,
//             NOTIFY=JIAYJ
```

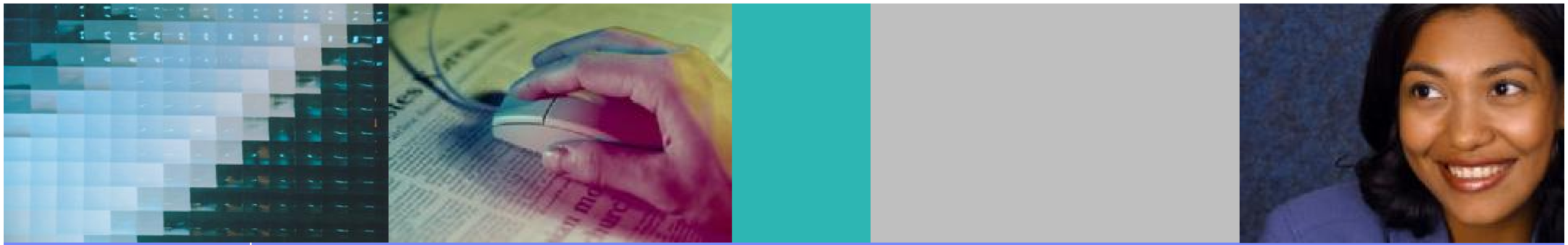


Table of contents

Procedures
Introduce

- ØFor Jobs that you run frequently or jobs that use the same JCL, precode job control statements into precedures
- ØProcedures consist of one or more complete steps
- ØEvery procedure must be given a name
- ØProcedures are invoked via the EXEC statement
- ØThree benefits of using procedures
  - ØSaves time by reducing the time required to code JCL
  - ØSaves library storage by eliminating duplicate JCL
  - ØReduces JCL errors by providing access to debugged JCL

Procedures

Introduce

There are two type of procedures

- When you place a procedure in the job input stream, it is called an in-stream procedure
  - Primarily used to test procedures
  - Reside in the job input stream and can be called only from that job stream

- A procedure cataloged in a library is called cataloged procedure
  - Can be called when needed
  - Resides in a procedure library

- Procedures cannot contain
  - JOB statement
  - DD * statement
  - DD DATA statement
  - Delimiter statement('/*' followed by 78 blanks)
  - Null statement('//' followed by 78 blanks)
  - NonJCL statements

Procedures

IN-STREAM procedure

```
//JOB1       JOB   MSGCLASS=A
//PROC1      PROC
//STEP       EXEC  PGM=PRINT
//IN         DD   DSN=PRTDATA,DISP=OLD
//OUT        DD   SYSOUT=*
//           PEND
//STEP1      EXEC  PROC=PROC1
//STEP2      EXEC  PROC1
```

- ØPlace an in-stream procedure in the input stream
  - ØAfter any JOB statement
  - ØBefore any EXEC statement that calls it

- ØAn in-stream procedure:
  - ØMust begin with a PROC statement
  - ØMust end with a PEND statement
  - ØIs called by an EXEC statement using the procedure name
  - ØMust be resubmitted each time the job is executer

Procedures

CATALOGING a procedure

After an in-stream procedure has been tested during execution, it can be cataloged. To use it, call it with an EXEC statement

- A procedure is said to be cataloged when it is placed in a procedure library(proclib)

- A proclib is a PDS and the member name is the procedure name coded on the calling EXEC statement.

- A procedure can be cataloged by placing it in one of three type of proclibs:
  - SYS1.PROCLIB – IBM supplied system procedure library
  - System proclibs – defined by an installation
  - A user-defined proclib

- Use the IEBUPDTE utility or ISPF to add a procedure to a proclib or modify a procedure

- A PROC statement can be included in a cataloged procedure

Procedures

PROCEDURE MODIFICATION

Two methods of modifying procedures

- Overriding,adding,or nullifying parameters
- Symbolic parameters

Either method can be used to modify

- In-stream and cataloged procedures
- EXEC,DD or output JCL statements

To make modifications to a procedure, you must know STEPNAMES and DDNAMES. You must also know what step the DDNAME is in

Procedures

Modifying EXEC statements

EXEC parameters can be overridden/added/nullified
Code mods on EXEC statement that invokes proc
Code mods for one proc step before the next proc step

```
PROC P1
//STEP1     EXEC  PGM=PAYROLL,TIME=(2,30),ACCT=1876
//STEP2     EXEC  PGM=PRINT,TIME=(4,30)

JOBSTREAM
//XY2    JOB
//STEPA      EXEC  P1,TIME.STEP1=(1,10),
//           ACCT.STEP1=,PARM.STEP2=TOP

RESULTING JCL
//STEP1     EXEC  PGM=PAYROLL,TIME=(1,10)
//STEP2     EXEC  PGM=PRINT,TIME=(4,30),PARM=TOP
```

Procedures

Modifying DD statements

DD parameters can be overridden/added/nullified
DD statements can be added after overriden
Nullify a keyword parm by coding ‘KEYWORD=‘

```
PROC P1
//STEP1      EXEC  PGM=PAYROLL
//A          DD  DSN=INPUT,DISP=OLD
//B          DD  DSN=OUTPUT,DISP=(,CATLG,DELETE),UNIT=3350,
//           SPACE=(CYL,(20,5)),DCB=(RECFM=FB,LRECL=80,BLKSIZE=320)
//STEP2      EXEC  PGM=PRINT
//A          DD DSN=OUTPUT,DISP=(OLD,DELETE),UNIT=3350,VOL=SER=PAK08
//B          DD SYSOUT=*

JOBSTREAM
//JOB       JOB MSGCLASS=A
//FS        EXEC  P1
//STEP1.A   DD DISP=(OLD,DELETE,DELETE)
//STEP1.B   DD UNIT=3390,DCB=(BLKSIZE=800)
//S1.D      DD *
           DATA
//S2.A      DD UNIT=,VOL=SER=,DISP=OLD
```

Procedures

Modifying DD statements

```
//JOB     JOB MSGCLASS=A
//STEP1      EXEC  PGM=PAYROLL
//A          DD  DSN=INPUT, DISP=(OLD,DELETE,DELETE)
//B          DD  DSN=OUTPUT,DISP=(,CATLG,DELETE),UNIT=3390,
//           SPACE=(CYL,(20,5)),DCB=(RECFM=FB,LRECL=80,BLKSIZE=800)
//D          DD *
           DATA
//STEP2      EXEC  PGM=PRINT
//A          DD DSN=OUTPUT,DISP=OLD      ,UNIT=3350,VOL=SER=PAK08
//B          DD SYSOUT=*
```

Procedures

Modifying Symbolic parameters

ØA sysbolic parameter is used to allow a JCL parameter to be easily changed or to be specifried at execution time

ØAny parameter, subparameter, or value in a procedure that can vary each time the procedure is called is a good symbolic paramter candidate

ØA symbolic parameter consists of an & followed by a name which is 1 to 7 alphanumeric or national charactors

ØValues assigned to the symbolic parameter via the PROC statement are de default value. A symbolic parameter value is assigned by coding the symbolic parameter, without the & and its value. These can appear in any order on the statement. Each apperance of the symbolic in the procedure will have this value assigned

ØThe procedure default values can be overridden by coding the symbolic parameter value on the EXEC statement that invokes the procedure.

Procedures

Modifying Symbolic parameters

```
PROC P1
//P1      PROC  UN=3390,OUT=OUTPUT
//STEP1     EXEC  PGM=PAYROLL
//A         DD  DSN=INPUT,DISP=OLD
//B         DD  DSN=&OUT,DISP=(,CATLG,DELETE),UNIT=&UN,
//          SPACE=(CYL,(20,5)),DCB=(RECFM=FB,LRECL=80,BLKSIZE=320)
//STEP2     EXEC  PGM=PRINT
//A         DD DSN=&OUT,DISP=(OLD,DELETE),UNIT=&UN,VOL=SER=PAK08
//B         DD SYSOUT=*
JOBSTREAM
//JOB      JOB MSGCLASS=A
//FS       EXEC  P1,UN=3380,OUT=TEST.OUT
RESULT è
//JOB      JOB MSGCLASS=A
//STEP1     EXEC  PGM=PAYROLL
//A         DD  DSN=INPUT,DISP=OLD
//B         DD  DSN=TEST.OUT,DISP=(,CATLG,DELETE),UNIT=3380,
//          SPACE=(CYL,(20,5)),DCB=(RECFM=FB,LRECL=80,BLKSIZE=320)
//STEP2     EXEC  PGM=PRINT
//A         DD DSN=TEST.OUT,DISP=(OLD,DELETE),UNIT=3380,VOL=SER=PAK08
//B         DD SYSOUT=*
```

EXERCISE 5

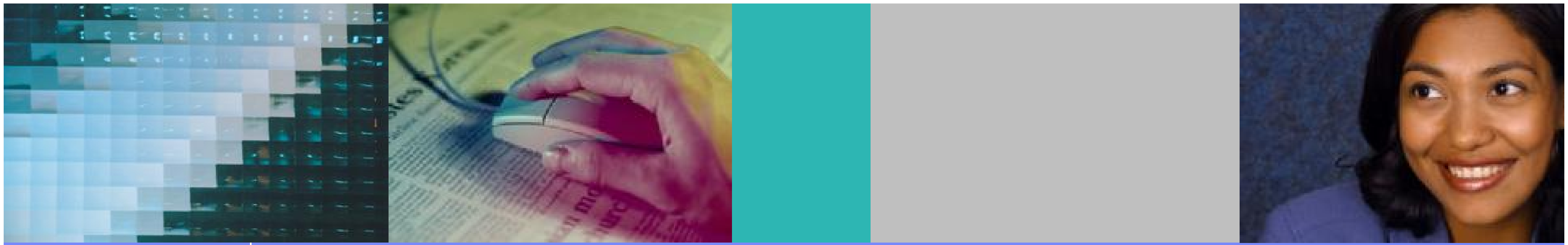


Table of contents

Advanced Topics

Conditional Execution of JOB Steps

Techniuqe 1: The COND Parameter

- ØSpecifies when a step should execute
- ØIs coded on the JOB or EXEC statement
- ØIs supported in every version/release of MVS

Techniuqe 2: The IF/THEN/ELSE/EDNIF statement

- ØSpecifies when a step should execute
- ØIs placed anywhere after the JOB statement
- ØIs supported only after MVS/ESA SP Ver 4

Advanced Topics
Relational Expressions

Relational Expressions are constructed from

ØNot operator **à** NOT

ØComparison Operators **à** GT,LT,NG,NL,EQ,NE,GE,LE

ØLogical Operators **à** AND, OR

ØRelational-Expressions Keywords **à** RC, ABEND, RUN, etc

Advanced Topics

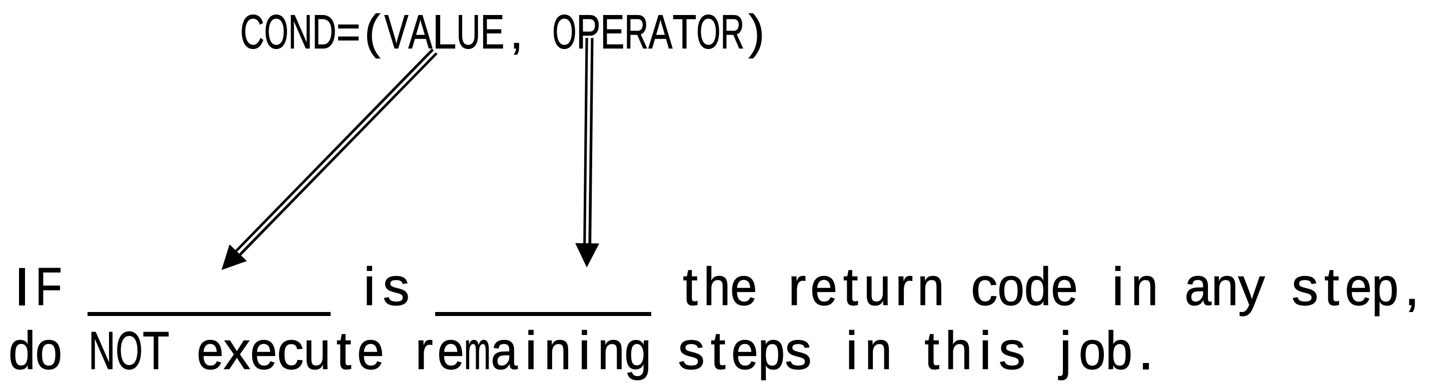
Return Code – Setting

```
//TEST JOB ……
//ST1  EXEC PGM=SORT
……
//ST2  EXEC PGM=PAYROLL
……
//ST3  EXEC PGM=PRINT
……
//ST4  EXEC PGM=CLEANUP
……
```

ØJob steps may be selectively bypassed based on the return code of a precding step or steps
ØPrograms assign a return code to signify a certain condition. If this condition occurs during execution, the program sets the return code
ØThe return code is not set if the step does not execute
ØReturn code for every step in the job are stored fro the life of the job
ØUse the COND parameter on the JOB and/or EXEC statements to test return code

Advanced Topics

COND parameter on the JOB statement

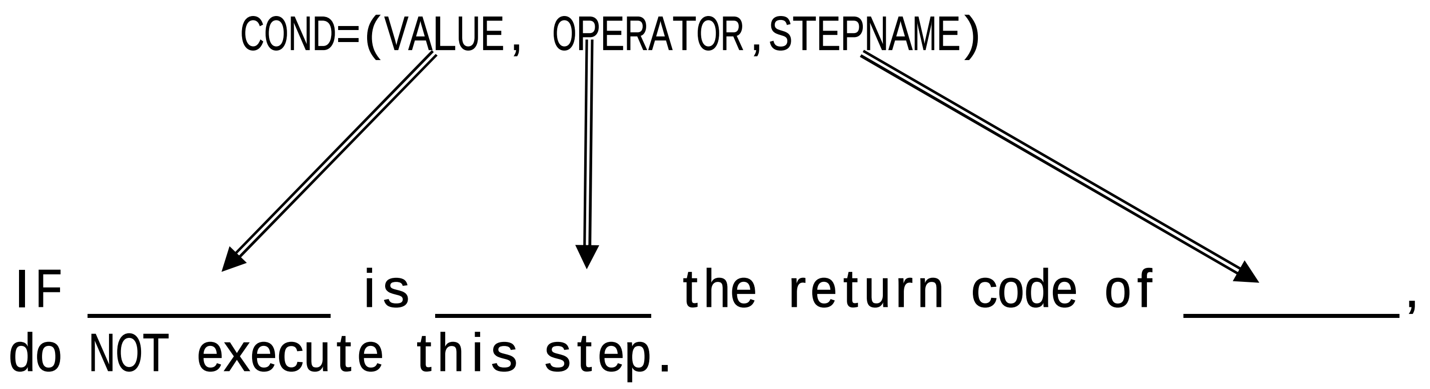


```
COND=(VALUE, OPERATOR)

IF __________ is _________ the return code in any step,
do NOT execute remaining steps in this job.
```

- ***Return code tests are made for each step in the job EXCEPT the first***
- ***If any condition tested is true, bypassed all remaining steps in the JOB***
- ***Return code tests specified on the JOB statement are performed before tests specified on the EXEC statement***

Advanced Topics

COND parameter on the EXEC statement



```
COND=(VALUE, OPERATOR,STEPNAME)

IF __________ is _________ the return code of _________,
do NOT execute this step.
```

- ***The return code test is performed on the step(s) specified***
- ***If no steps are specified, the test is performed on all preceding steps***
- ***If any condition tested is true, bypass the testing jobstep***

Advanced Topics

Abnormal Termination Testing

```
COND=EVEN
Execute this step even if a prior step abended

COND=ONLY
Execute this step only if a prior step abended
```

ØBypassing a step because of a return code test is not the same as Abnormally Terminating(ABEND) the step. The system ABENDS a step following a serious error that prevents proper execution. Bypassing a step is simply omitting the step
ØIf a steps ABENDS, the system bypasses all following steps in the job

Advanced Topics

COND Testing Example

```
//TEST JOB ······
//ST1  EXEC PGM=SORT
······
//ST2  EXEC PGM=PAYROLL,COND=(8,LE,ST1)
······
//ST3  EXEC PGM=PRINT,COND=(8,LE,ST1),(12,EQ,ST2)
······
//ST4  EXEC PGM=CLEANUP,COND=((20,EQ),EVEN)
······
```

RC=0

RC=12

NO RC

RC=0

Advanced Topics

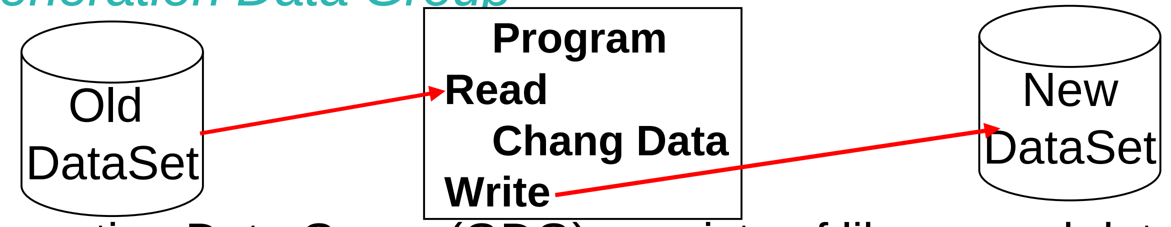
IF/THEN/ELSE/ENDIF statement construct

```
//name1 IF relational expressions   THEN
……
//name2 ELSE
……
//name3 ENDIF
```

```
//SAMPLE  JOB
//STEP1   EXEC  PGM=PROG1
……
//CHECK   IF (RC > 0 | ABEND) THEN
//        ELSE
//GOODRUN EXEC  PGM=ANALYZE
……
//SAVE    EXEC  PGM=SAVEDATA
……
//ENDCHK  ENDIF
//NXSTEP  EXEC PGM=CONTINUE
```

Advanced Topics

GDG – Generation Data Group



- A Generation Data Group (GDG) consists of like named data sets that are chronologically or functionally related. A data set in a GDG is called a generation

- Why use Generation Data Group (GDG)?
  - ü CASE 1. You attempt to catalog a new data set with the same name as an existing data set.
    CATALOG ERROR1 NAME ALREADY IN THE CATLOG
  - ü CASE 2. You keep the new data set with the same name
    JCL ERROR! DUPLICATE NAME ON VTOC
  - ü CASE 3. You catalog or keep the data set with a different name.
    JCL STATEMENTS MUST BE CHANGED TO THE NEW NAMES

Advanced Topics

GDG – Base Catalog Entry

```
TEST.GDG  LIMIT(7) NOEMPTY NOSCRATCH FOR (14) PASSWORDS
```

Before generation data sets can be created, a GDG Base must be defined, use IDCAMS to define:

•LIMIT	Maximum number of Generations allowed for this GDG entry
•EMPTY	When limit exceed, uncatalog all generations
•NOEMPTY	When limit exceed, uncatalog oldest entry only
•SCRATCH	Delete the DSCB for any entry uncataloged
•NOSCRATCH	Do not delete the DSCB for any entry uncataloged
•FOR	Retention period
•TO	Expiration Date
•OWNER	User information (Up to 8 charactors)
•Passwords	Depend on protection mechanism used

Advanced Topics

GDG – DSNAME Specification

Relative Data Set Name

DSN=____._____._____(±n)

(+n) - Add a new Generation
(+0) – Use Current Generation
(-n) – Use an old Genereation

Example:
DSN=FIRST.GDG(+1)

Absolute Data Set Name

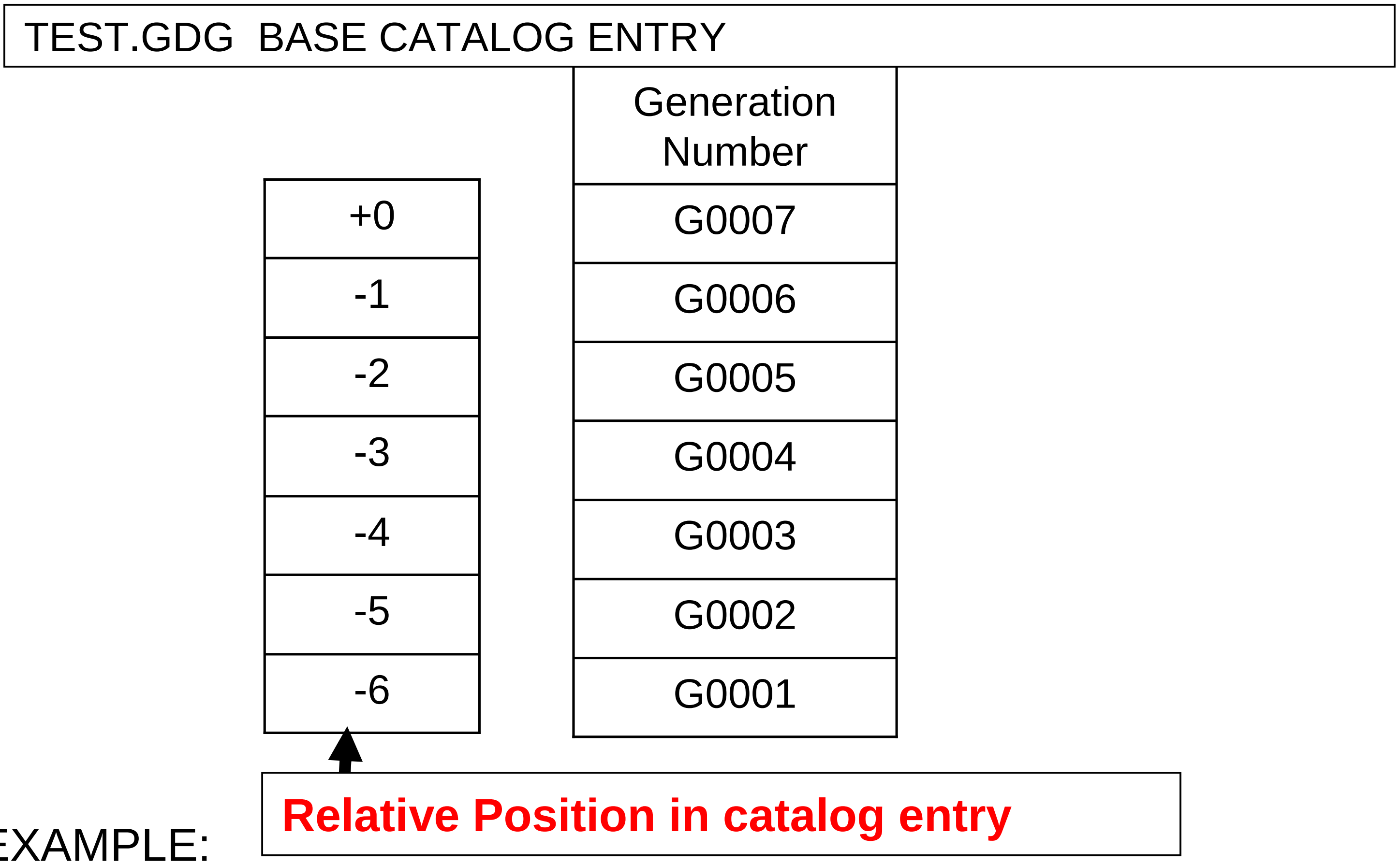
DSN=____._____._____.GxxxxVyy

GxxxxVyy:
xxxx – Generation Number
yy – Version Number

Example:
DSN=FIRST.GDG.G0052V00

Advanced Topics

GDG – Catalog Entry



TEST.GDG BASE CATALOG ENTRY

	Generation Number
+0	G0007
-1	G0006
-2	G0005
-3	G0004
-4	G0003
-5	G0002
-6	G0001

Relative Position in catalog entry

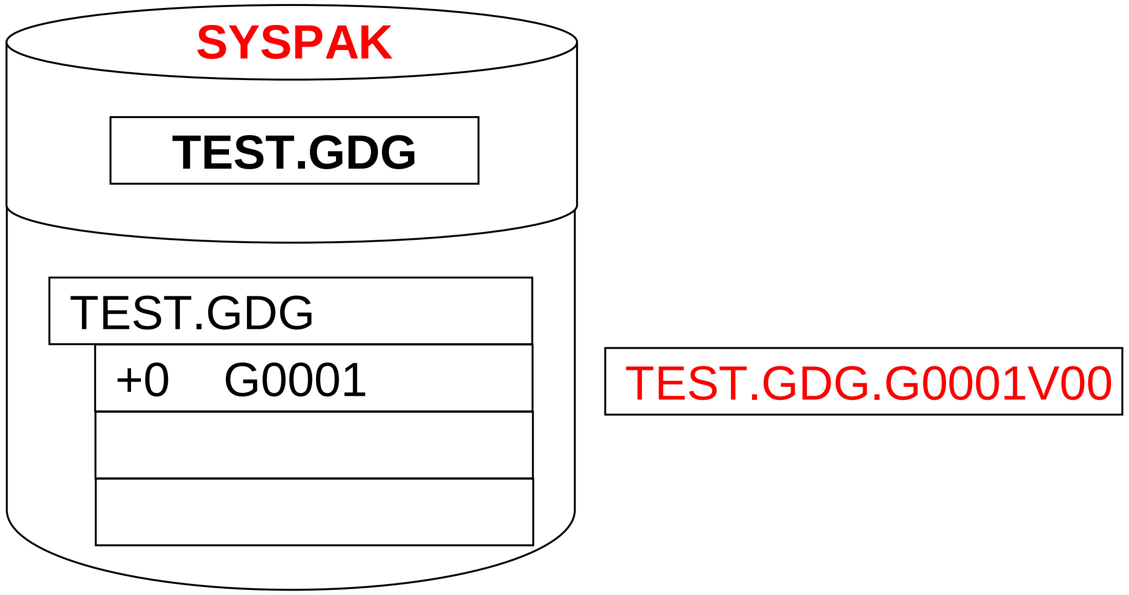
EXAMPLE:

```
//DD1   DD  DSN=TEST.GDG(=0),DISP=OLD      LOCATE CURRENT (TOP) ENTRY
//DD2   DD  DSN=TEST.GDG(-6),DISP=OLD       LOCATE OLDEST ENTRY
```

Advanced Topics

GDG Example – First Generation

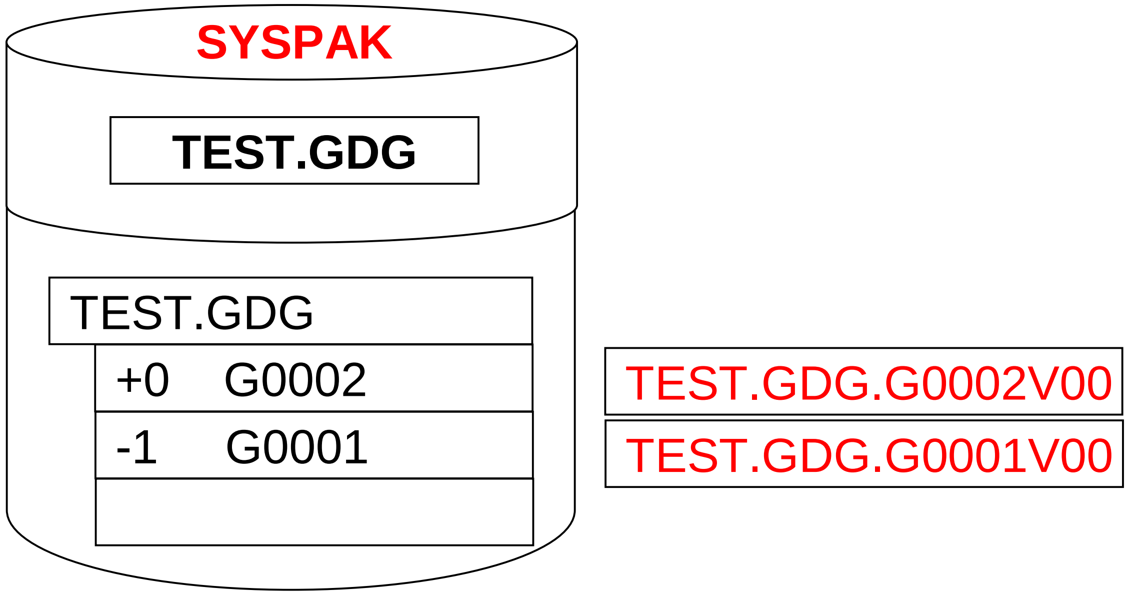
```
//EXAMPLE JOB   378,SMITH,CLASS=T
//STEP1   EXEC  PGM=USERPGM1
//FIRST   DD    DSN=TEST.GDG(+1),DISP=(,CATLG,DELETE),
//        SPACE=(CYL,(40,5),RLSE),UNIT=SYSDA
//INPUT   DD    DSN=INITIAL.DATA,DISP=OLD
```



Advanced Topics

GDG Example – Second Generation

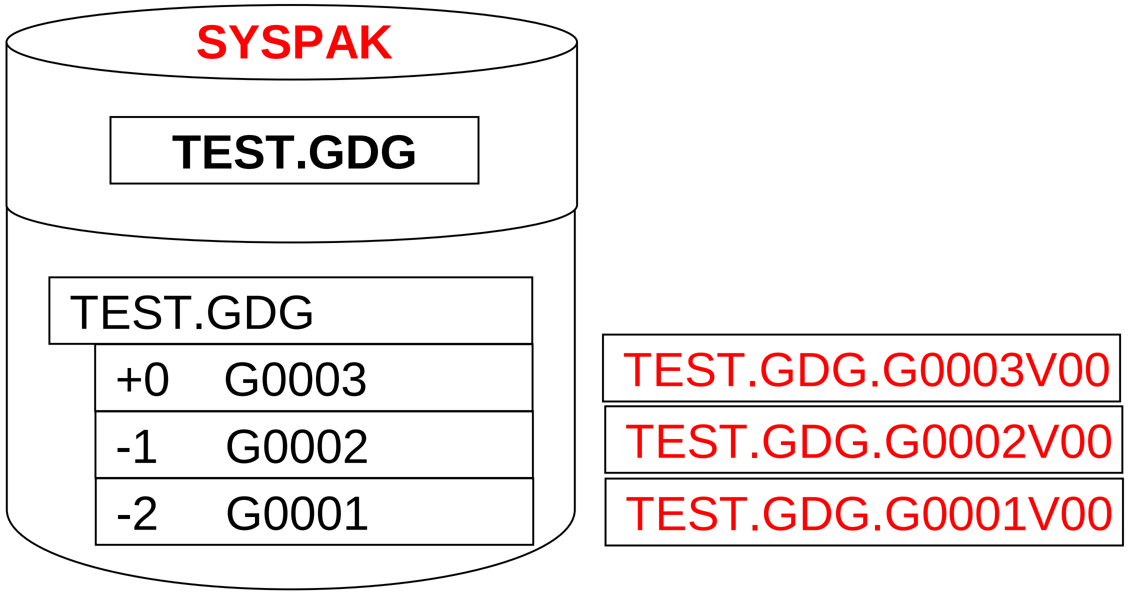
```
//EXAMPLE2 JOB   378,SMITH,CLASS=G
//STEP1    EXEC  PGM=MAINLINE
//GDGIN    DD    DSN=TEST.GDG(+0),DISP=OLD
//GDGOUT   DD    DSN=TEST.GDG(+1),DISP=(NEW,CATLG,DELETE),
//         SPACE=(CYL,(40,5),RLSE),UNIT=SYSDA
```



Advanced Topics

GDG Example – Third Generation

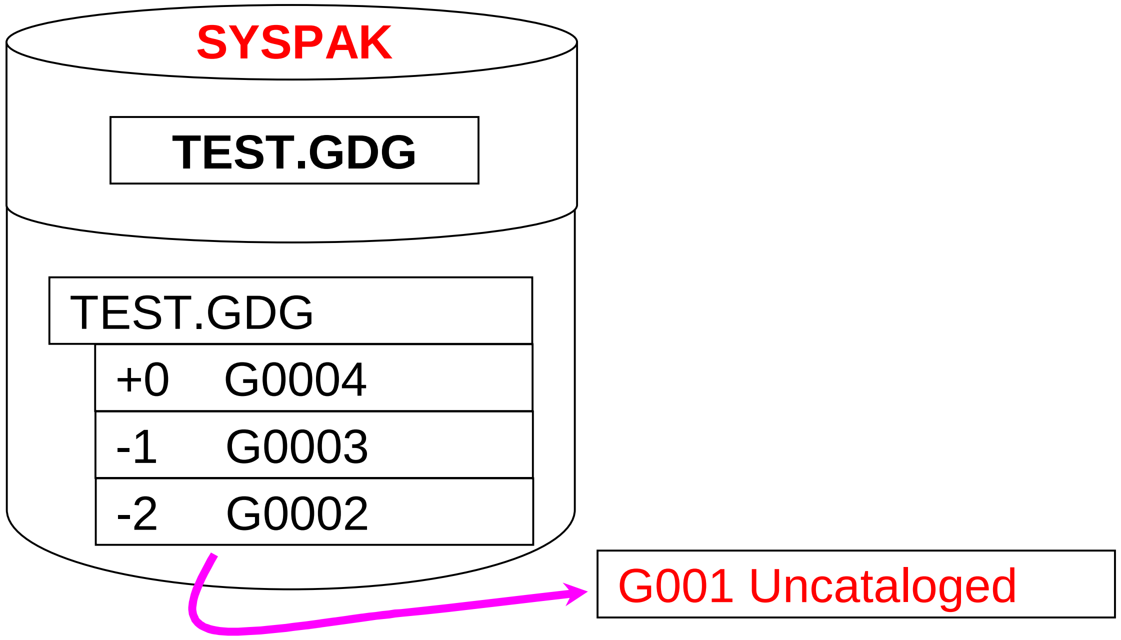
```
//EXAMPLE2 JOB   378,SMITH,CLASS=G
//STEP1    EXEC  PGM=MAINLINE
//GDGIN    DD    DSN=TEST.GDG(+0),DISP=OLD
//GDGOUT   DD    DSN=TEST.GDG(+1),DISP=(NEW,CATLG,DELETE),
//         SPACE=(CYL,(40,5),RLSE),UNIT=SYSDA
```



Advanced Topics

GDG Example – Limit Exceeded

```
//EXAMPLE2 JOB   378,SMITH,CLASS=G
//STEP1    EXEC  PGM=MAINLINE
//GDGIN    DD    DSN=TEST.GDG(+0),DISP=OLD
//GDGOUT   DD    DSN=TEST.GDG(+1),DISP=(NEW,CATLG,DELETE),
//         SPACE=(CYL,(40,5),RLSE),UNIT=SYSDA
```



Advanced Topics
GDG – DEFINE GENERATIONDATAGROUP

The DEFINE GENERATIONDATAGROUP command creates a catalog entry for a GDG. The syntax of this command is:

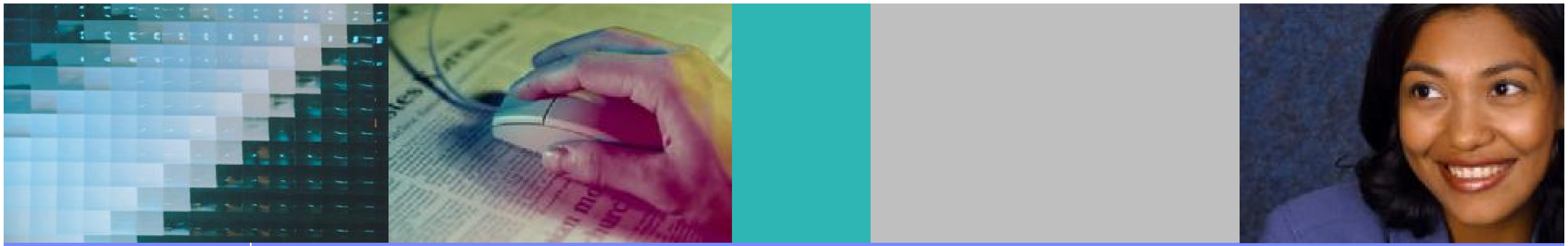
DEFINE	GENERATIONDATAGROUP (NAME(*entryname*) LIMIT(*limit*) [EMPTY|NOEMPTY] [OWNER(*ownerid*)] [SCRATCH|NOSCRATCH] [TO(*date*)|FOR(*days*)] [CATALOG(*catname*[*/password*])]

Advanced Topics

GDG – DEFINE GENERATIONDATAGROUP sample

```
//DEFGDG1  JOB   ···
//STEP1    EXEC  PGM=IDCAMS
//GDGM00   DD    DSN=GDG01,DISP=(,KEEP)
//         SPACE=(TRK,(0)),UNIT=DISK,VOL=SER=VSER03,
//         DCB=(RECFM=FB,BLKSIZE=2000,LRECL=100)
//SYSPRINT DD    SYSOUT=A
//SYSIN    DD    *
     DEFINE GENERATIONDATAGROUP  -
     (NAME(GDG01)   -
      EMPTY –
      NOSCRATCH –
      LIMIT(255)
/*
```

EXERCISE 6



ISSC Shanghai, AMS, GCG

Any Existing Process Could Be Improved!

Thanks very much!