# Breaking Down Datasets and the DD Statement



**Dave Nicolette**Software Developer

@davenicolette neopragma.com

# Module Intro and Overview

## Overview



- Datasets
- Access Methods
- Associating datasets with job steps
- DD statement scope
- DD statement concatenation

# DASD, Data Sets, and Access Methods

# "Data Set," "DD Name," "File"

//CUSTF DD DSN=CUSTOMER.INFO.FILE

SELECT CUSTOMER-INFO...

Data Set

DD Name

File

```
//FILE1 DD DSN=HIGH.MEDIUM.LOW,
// DISP=(MOD, DELETE, DELETE),
// SPACE=(TRK,1)

//FILE2 DD DSN=SOME.THING.HERE,
// DISP=(NEW, CATLG, DELETE),
// SPACE=(CYL,20)
```



```
//FILE1 DD DSN=HIGH.MEDIUM.LOW,
// DISP=(MOD, DELETE, DELETE),
// SPACE=(TRK,1)

//FILE2 DD DSN=SOME.THING.HERE,
// DISP=(NEW, CATLG, DELETE),
// SPACE=(CYL,20)
```

#### ➤ Track



```
//FILE1 DD DSN=HIGH.MEDIUM.LOW,
                                               Track
        DISP=(MOD, DELETE, DELETE),
//
        SPACE=(TRK, 1)
//
//FILE2 DD DSN=SOME.THING.HERE,
        DISP=(NEW, CATLG, DELETE),
//
        SPACE=(CYL, 20)
//
                                              Cylinder
```

```
//FILE1 DD DSN=HIGH.MEDIUM.LOW,
// DISP=(MOD, DELETE, DELETE),
// SPACE=(TRK,1)

//FILE2 DD DSN=SOME.THING.HERE,
// DISP=(NEW, CATLG, DELETE),
// SPACE=(CYL,20)
```

#### Where are the tracks?



Where are the cylinders?

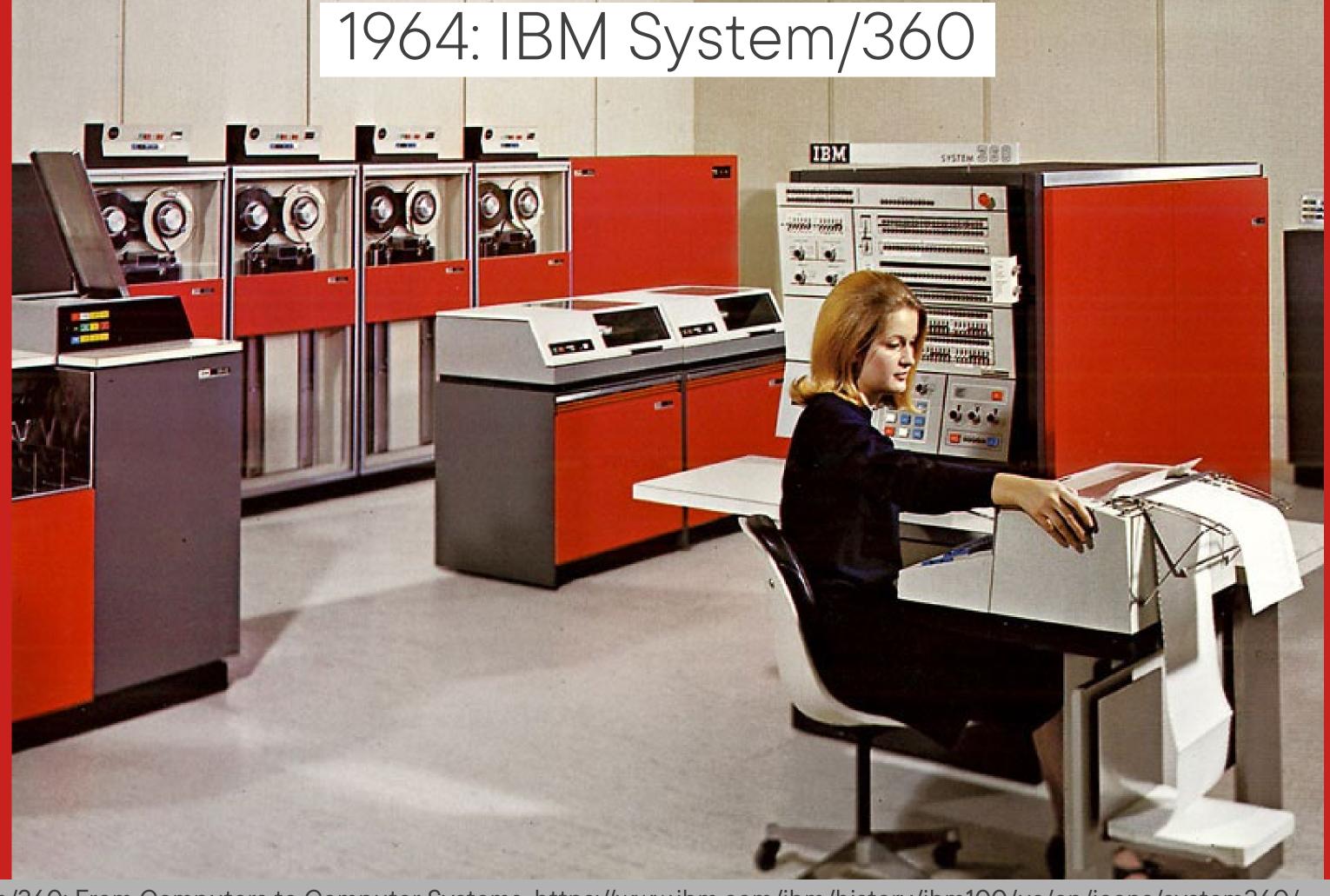
# System Z Design Goals

# **Backward Compatibility**

The system can run any executable compiled on the same family of computers since 1964

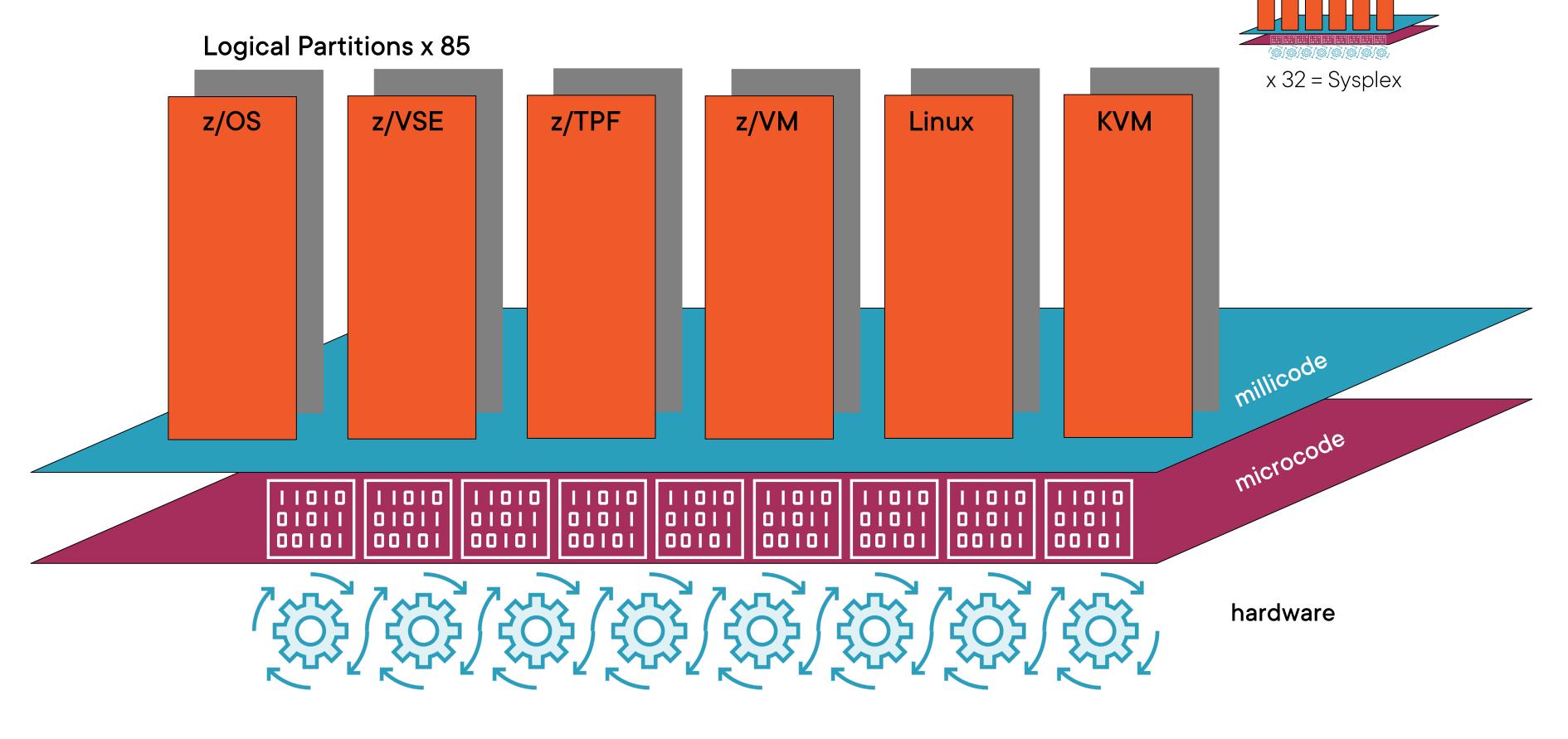
## Modernization

The system can exploit any new computing technologies developed since 1964.

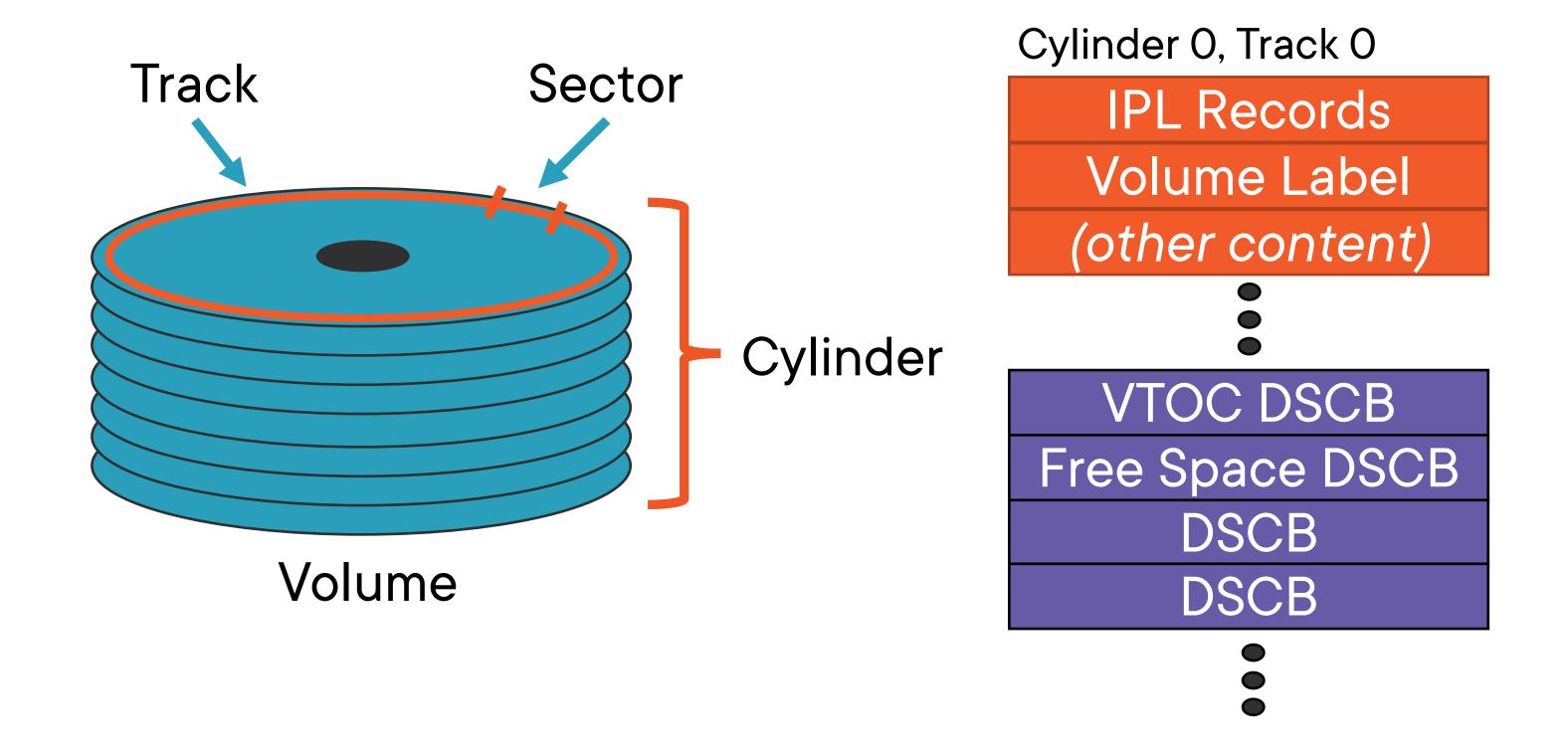


System/360: From Computers to Computer Systems, https://www.ibm.com/ibm/history/ibm100/us/en/icons/system360/

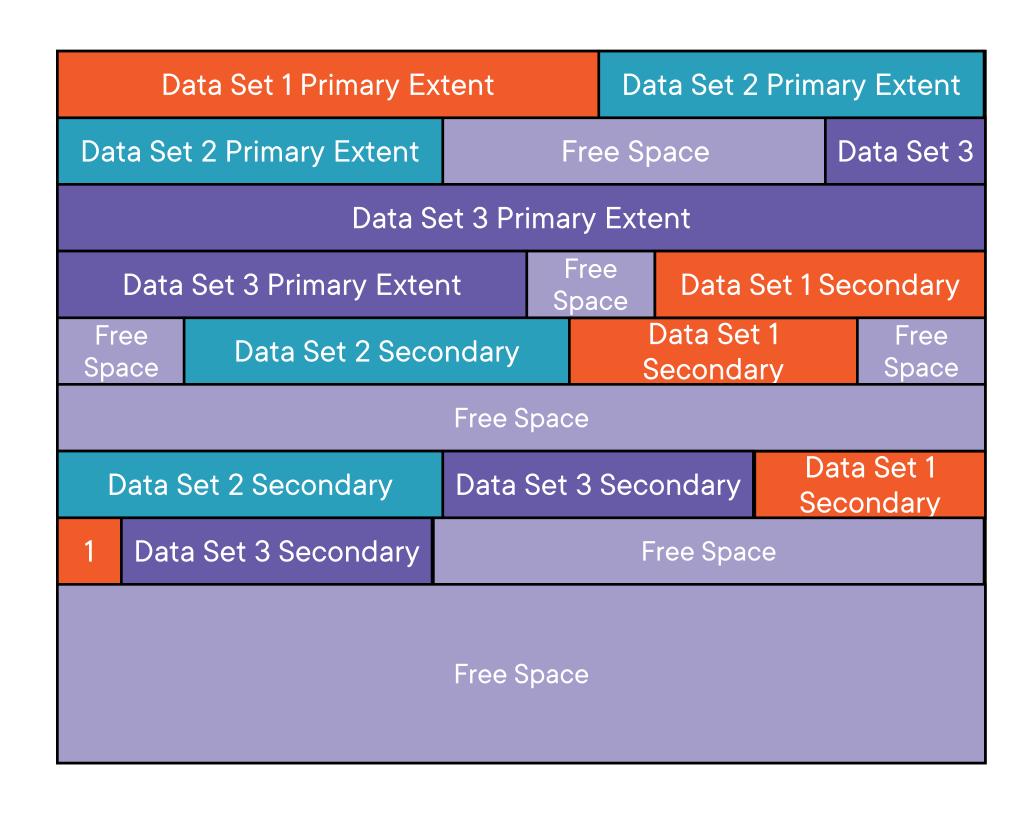
# System Z Virtualization



# DASD - Direct Access Storage Device



# Primary and Secondary Extents



# Data Set Types and Access Methods

## Overview



- Data Set Types
- Access Methods
- Most Frequently-used Data Set Types
- Rules for Data Set Names
- Data Set Status and Disposition

# Access Method

z/OS system software that knows how to access, modify, and manage a particular type of data set

# Most Frequently-used Data Set Types

#### **QSAM**

Queued Sequential Access Method

#### GDG

Generation Data Group (GDG)

#### **BPAM**

Basic Partitioned Access Method

#### **VSAM**

Virtual Sequential Access Method

#### **HFS File**

POSIX file (Unix System Services)

```
1 2 3 4 5 6 7 8
12345678901234567890123456789012345678901234567890123456789012345678901234567890123456789012345678901234567890123456789012345678901234567890123456789012345678901234567890123456789012345678901234567890123456789012345678901234567890123456789012345678901234567890123456789012345678901234567890123456789012345678901234567890123456789012345678901234567890123456789012345678901234567890123456789012345678901234567890123456789012345678901234567890123456789012345678901234567890123456789012345678901234567890123456789012345678901234567890123456789012345678901234567890123456789012345678901234567890123456789012345678901234567890123456789012345678901234567890123456789012345678901234567890123456789012345678901234567890123456789012345678901234567890123456789012345678901234567890123456789012345678901234567890123456789012345678901234567890123456789012345678901234567890123456789012345678901234567890123456789012345678901234567890123456789012345678901234567890123456789012345678901234567890123456789012345678901234567890123456789012345678901234567890123456789012345678901234567890123456789012345678901234567890123456789012345678901234567890123456789012345678901234567890123456789012345678901234567890123456789012345678901234567890123456789012345678901234567890123456789012345678901234567890123456789012345678901234567890123456789012345678901234567890123456789012345678901234567890123456789012345678901234567890123456789012345678901234567890123456789012345678901234567890123456789012345678901234567890123456789012345678901234567890123456789012345678901234567890123456789012345678901234567890123456789012345678901234567890123456789012345678901234567890123456789012345678901234567890123456789012345678901234567890123456789012345678901234567890123456789012345678901234567890123456789012345678901234567890123456789012345678901234567890123456789012345678901234567890123456789012345678901234567890123456789012345678901234567890123456789012345678901234567890123456789012345678901234567890123456789012345678901234567890123456789012345678901234567890
```

## Unqualified Data Set Name

Between 1 and 8 characters selected from alphanumeric, national (\$, #, @), the hyphen (-) and the character represented by X'C0'.

```
12345678901234567890123456789012345678901234567890123456789012345678901234567890
             DSN=ALPHA.ONE
                                                        9 CHARS, OK
//DD1
        DD
             DSNAME=EVERYONE.LIKES.CHEESE
//DD2
        DD
                                                        21 CHARS, OK
        DD
             DSN=AMBER#45.TEST.COBOL.SOURCE
                                                        26 CHARS, OK
//DD3
            DSN=ACCT.MONTHLY.REPORTS.JANUARY.2022(0)
//DD4
       DD
                                                        GDG 33 CHARS, OK
             DSN=##TG@313.OBVIOUSLY.WILL.NOT.WORK
                                                        "OBVIOUSLY" > 8
//DD5
       DD
//* Next DD statement - name is longer than 44 characters
            DSN=THIS.NAME.IS.DEFIN.ITELY.TOO.LONG.TO.WORK.PROPER.LY
//DD6
// . . .
```

#### **Qualified Data Set Name**

Multiple unqualified names connected by periods. Length must not exceed 44 characters, including periods

# DD DISP Parameter

```
{DISP=[status]}
{DISP=[status][,normal-disp][,abnormal-disp]}
DISP=( [NEW]
                    [, DELETE]
                                  [, DELETE] )
                    [, KEEP]
       [OLD]
                                  [, KEEP]
       [SHR|SHARE] [,PASS] [,CATLG]
       [MOD]
                   [, CATLG] [, UNCATLG]
                    [, UNCATLG]
```

```
1 2 3 4 5 6 7 8

123456789012345678901234567890123456789012345678901234567890123456789012345678901234567890

//DD1 DD DSN=..., DISP=NEW

//DD1 DD DSN=..., DISP=(NEW, DELETE, DELETE) <= same

//*

//DD2 DD DSN=..., DISP=(NEW, DELETE)

//DD2 DD DSN=..., DISP=(NEW, DELETE, DELETE) <= same

//*

//DD3 DD DSN=..., DISP=(NEW, KEEP)

//DD3 DD DSN=..., DISP=(NEW, KEEP, KEEP) <= same
```

# DD DISP Examples (1)

For status NEW, in most cases the default abnormal disposition is the same as the normal disposition.

```
1 2 3 4 5 6 7 8

1234567890123456789012345678901234567890123456789012345678901234567890123456789012345678901234567890123456789012345678901234567890123456789012345678901234567890123456789012345678901234567890123456789012345678901234567890123456789012345678901234567890123456789012345678901234567890123456789012345678901234567890123456789012345678901234567890123456789012345678901234567890123456789012345678901234567890123456789012345678901234567890123456789012345678901234567890123456789012345678901234567890123456789012345678901234567890123456789012345678901234567890123456789012345678901234567890123456789012345678901234567890123456789012345678901234567890123456789012345678901234567890123456789012345678901234567890123456789012345678901234567890123456789012345678901234567890123456789012345678901234567890123456789012345678901234567890123456789012345678901234567890123456789012345678901234567890123456789012345678901234567890123456789012345678901234567890123456789012345678901234567890123456789012345678901234567890123456789012345678901234567890123456789012345678901234567890123456789012345678901234567890123456789012345678901234567890123456789012345678901234567890123456789012345678901234567890123456789012345678901234567890123456789012345678901234567890123456789012345678901234567890123456789012345678901234567890123456789012345678901234567890123456789012345678901234567890123456789012345678901234567890123456789012345678901234567890123456789012345678901234567890123456789012345678901234567890123456789012345678901234567890123456789012345678901234567890123456789012345678901234567890123456789012345678901234567890123456789012345678901234567890123456789012345678901234567890123456789012345678901234567890123456789012345678901234567890123456789012345678901234567890123456789012345678901234567890123456789012345678901234567890123456789012345678901234567890123456789012345678901234567890123456789012345678901234567890123456789012345678901234567890123456789012345678901234567890123456789012345678901234567890123456789012345678901234567890123456789
```

# DD DISP Examples (2)

For status NEW, in most cases the default abnormal disposition is the same as the normal disposition.

```
1 2 3 4 5 6 7 8

1234567890123456789012345678901234567890123456789012345678901234567890

//DD7 DD DSN=..., DISP=(NEW, PASS, DELETE)

//DD8 DD DSN=..., DISP=(NEW, PASS, KEEP)

//DD9 DD DSN=..., DISP=(NEW, PASS, CATLG)

//DD10 DD DSN=..., DISP=(NEW, PASS, UNCATLG)
```

## DD DISP Examples (3)

For status NEW and normal disposition PASS, if all job steps terminate normally (condition code zero) the data set is deleted at the end of the last job step.

If any steps fail, the value of the third subparameter is honored.

```
1 2 3 4 5 6 7 8

1234567890123456789012345678901234567890123456789012345678901234567890123456789012345678901234567890123456789012345678901234567890123456789012345678901234567890123456789012345678901234567890123456789012345678901234567890123456789012345678901234567890123456789012345678901234567890123456789012345678901234567890123456789012345678901234567890123456789012345678901234567890123456789012345678901234567890123456789012345678901234567890123456789012345678901234567890123456789012345678901234567890123456789012345678901234567890123456789012345678901234567890123456789012345678901234567890123456789012345678901234567890123456789012345678901234567890123456789012345678901234567890123456789012345678901234567890123456789012345678901234567890123456789012345678901234567890123456789012345678901234567890123456789012345678901234567890123456789012345678901234567890123456789012345678901234567890123456789012345678901234567890123456789012345678901234567890123456789012345678901234567890123456789012345678901234567890123456789012345678901234567890123456789012345678901234567890123456789012345678901234567890123456789012345678901234567890123456789012345678901234567890123456789012345678901234567890123456789012345678901234567890123456789012345678901234567890123456789012345678901234567890123456789012345678901234567890123456789012345678901234567890123456789012345678901234567890123456789012345678901234567890123456789012345678901234567890123456789012345678901234567890123456789012345678901234567890123456789012345678901234567890123456789012345678901234567890123456789012345678901234567890123456789012345678901234567890123456789012345678901234567890123456789012345678901234567890123456789012345678901234567890123456789012345678901234567890123456789012345678901234567890123456789012345678901234567890123456789012345678901234567890123456789012345678901234567890123456789012345678901234567890123456789012345678901234567890123456789012345678901234567890123456789012345678901234567890123456789012345678901234567890123456789012345678901234567890123456789
```

## DD DISP Examples (4)

For existing datasets, the default disposition is KEEP.

If normal disposition is coded, the default abnormal disposition is the same as the normal disposition. There are special cases – out of scope for this course.

```
1 2 3 4 5 6 7 8

1234567890123456789012345678901234567890123456789012345678901234567890123456789012345678901234567890123456789012345678901234567890123456789012345678901234567890123456789012345678901234567890123456789012345678901234567890123456789012345678901234567890123456789012345678901234567890123456789012345678901234567890123456789012345678901234567890123456789012345678901234567890123456789012345678901234567890123456789012345678901234567890123456789012345678901234567890123456789012345678901234567890123456789012345678901234567890123456789012345678901234567890123456789012345678901234567890123456789012345678901234567890123456789012345678901234567890123456789012345678901234567890123456789012345678901234567890123456789012345678901234567890123456789012345678901234567890123456789012345678901234567890123456789012345678901234567890123456789012345678901234567890123456789012345678901234567890123456789012345678901234567890123456789012345678901234567890123456789012345678901234567890123456789012345678901234567890123456789012345678901234567890123456789012345678901234567890123456789012345678901234567890123456789012345678901234567890123456789012345678901234567890123456789012345678901234567890123456789012345678901234567890123456789012345678901234567890123456789012345678901234567890123456789012345678901234567890123456789012345678901234567890123456789012345678901234567890123456789012345678901234567890123456789012345678901234567890123456789012345678901234567890123456789012345678901234567890123456789012345678901234567890123456789012345678901234567890123456789012345678901234567890123456789012345678901234567890123456789012345678901234567890123456789012345678901234567890123456789012345678901234567890123456789012345678901234567890123456789012345678901234567890123456789012345678901234567890123456789012345678901234567890123456789012345678901234567890123456789012345678901234567890123456789012345678901234567890123456789012345678901234567890123456789012345678901234567890123456789012345678901234567890123456789012345678901234567890123456789
```

## DD DISP Examples (5)

For existing datasets, the default disposition is KEEP.

If normal disposition is coded, the default abnormal disposition is the same as the normal disposition.

There are special cases – out of scope for this course.

```
1 2 3 4 5 6 7 8

1234567890123456789012345678901234567890123456789012345678901234567890123456789012345678901234567890123456789012345678901234567890123456789012345678901234567890123456789012345678901234567890123456789012345678901234567890123456789012345678901234567890123456789012345678901234567890123456789012345678901234567890123456789012345678901234567890123456789012345678901234567890123456789012345678901234567890123456789012345678901234567890123456789012345678901234567890123456789012345678901234567890123456789012345678901234567890123456789012345678901234567890123456789012345678901234567890123456789012345678901234567890123456789012345678901234567890123456789012345678901234567890123456789012345678901234567890123456789012345678901234567890123456789012345678901234567890123456789012345678901234567890123456789012345678901234567890123456789012345678901234567890123456789012345678901234567890123456789012345678901234567890123456789012345678901234567890123456789012345678901234567890123456789012345678901234567890123456789012345678901234567890123456789012345678901234567890123456789012345678901234567890123456789012345678901234567890123456789012345678901234567890123456789012345678901234567890123456789012345678901234567890123456789012345678901234567890123456789012345678901234567890123456789012345678901234567890123456789012345678901234567890123456789012345678901234567890123456789012345678901234567890123456789012345678901234567890123456789012345678901234567890123456789012345678901234567890123456789012345678901234567890123456789012345678901234567890123456789012345678901234567890123456789012345678901234567890123456789012345678901234567890123456789012345678901234567890123456789012345678901234567890123456789012345678901234567890123456789012345678901234567890123456789012345678901234567890123456789012345678901234567890123456789012345678901234567890123456789012345678901234567890123456789012345678901234567890123456789012345678901234567890123456789012345678901234567890123456789012345678901234567890123456789012345678901234567890123456789
```

## DD DISP Examples (6) – Default Values

DD17. If DISP is omitted altogether, the default is (NEW,DELETE,DELETE).

DD18. If the status subparameter is omitted, the default is NEW.

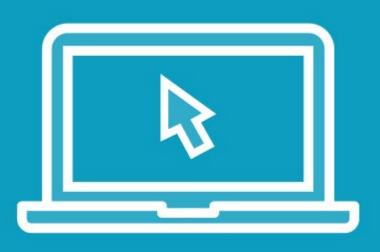
DD19. For existing datasets, the default disposition is KEEP.

```
DD DSN=MY.NEW.DATA.SET,
//DD1
         DISP=(NEW, CATLG, DELETE),
//
//
         DSORG=PS,
         SPACE=(TRK, 1)
      DD DSN=MY.NEW.DATA.SET,
//DD1
         DISP=(NEW, CATLG, DELETE),
         DSORG=PSU,
         SPACE=(TRK, 1)
```

◆ PS stands for Physical Sequential. It pertains to QSAM, BSAM, and other types not covered in the course. You will use this a lot in your work.

■ PSU stands for Physical Sequential
Unmoveable. It pertains to datasets that
cannot be relocated from their position on a
volume. This is out of scope for the course.

## Demo



- Try different values for the DISP parameter of the DD statement for QSAM data sets
  - -Why? To observe the behavior of various normal and abnormal disposition settings.
  - How? We'll use the do-nothing utility, IEFBR14, and a small test program that sets the condition code to a value we pass in as a PARM.



- Craftspeople often build a rig or template or frame to help them fashion parts consistently for the things they're building.
- Software people do the same thing. To explore the behaviors of different values of the DD disposition parameter, we're going to use a sort of "rig" made of software to force job steps to fail.
- It's a program called SETCC, which you can find in your course handouts. You can upload and assemble the program on your own Z system to duplicate the demo.

# Record Formats and DD Coding for QSAM

```
//DD1 DD DSN=MY.NEW.DATA.SET,
// DISP=(NEW,CATLG,DELETE),
// SPACE=(CYL,30),
// DSORG=PS...

//DD1 DD DSN=MY.NEW.DATA.SET,
// DISP=(NEW,CATLG,DELETE),
// SPACE=(CYL,30),
// DSORG=PSU...
```

■ Dataset organization physical sequential applies to QSAM and other sequential data set types.

■ Dataset organization physical sequential unmoveable means the dataset cannot be relocated from its location on DASD. This is out of scope for the course.

# DD Statement Parameters for QSAM

#### **SPACE**

How to define the size of primary and secondary extents

#### **DISP**

How to specify the level of control needed by the step, and what to do with the data set after the step

#### **DSORG**

How to specify the organization of the data set

#### **DSNAME**

The rules for valid data set names

#### **RECFM**

The record format of the data set

# Three Key Characteristics of Data Sets

DD DSORG=...

Data Set
Organization

DD RECFM=...

Fixed or Variable
Length
Records

DD RECFM=...

Blocked or Unblocked Records

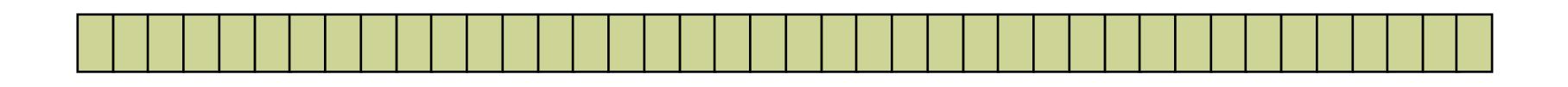
# Concept: Blocks and Logical Records

### **Block**

The unit of data transfer between DASD and programs

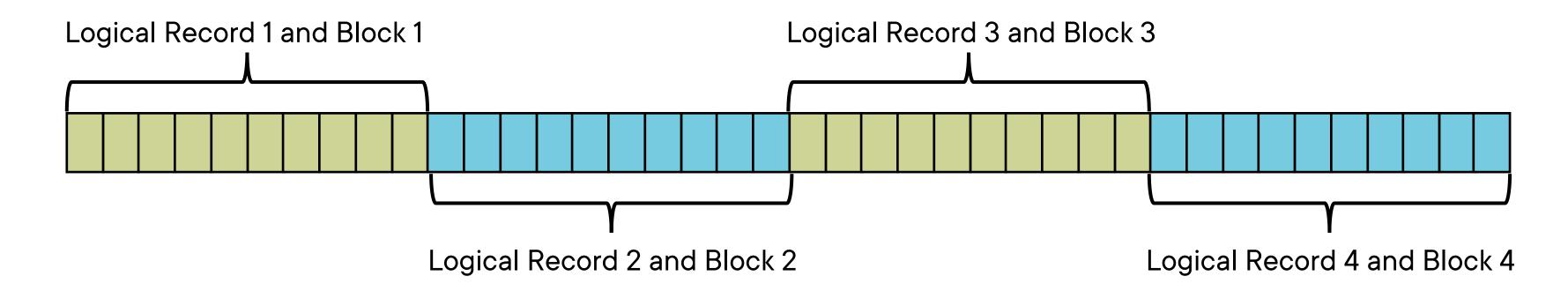
# Logical Record

The logical unit of content in a data set



## Fixed-length Records, Unblocked

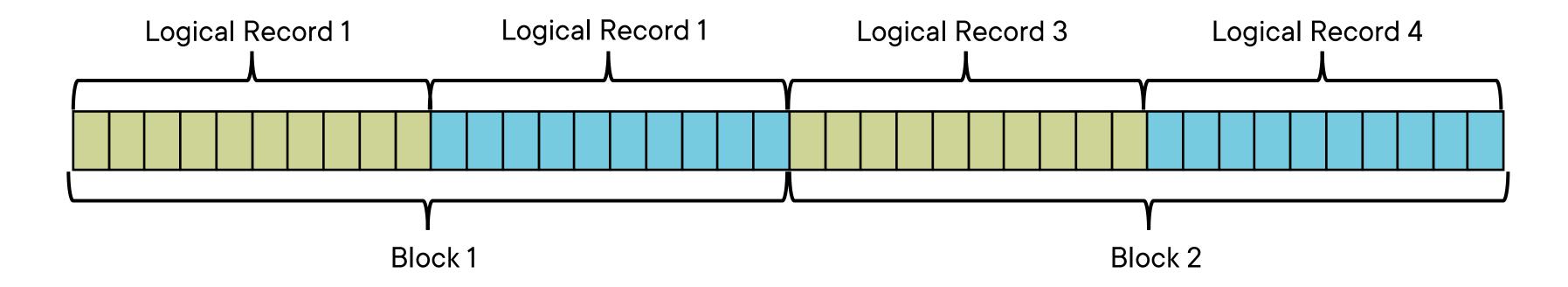
DD RECFM=F, LRECL=10, BLKSIZE=10



Rule: Block size must equal logical record length

# Fixed-length Records, Blocked

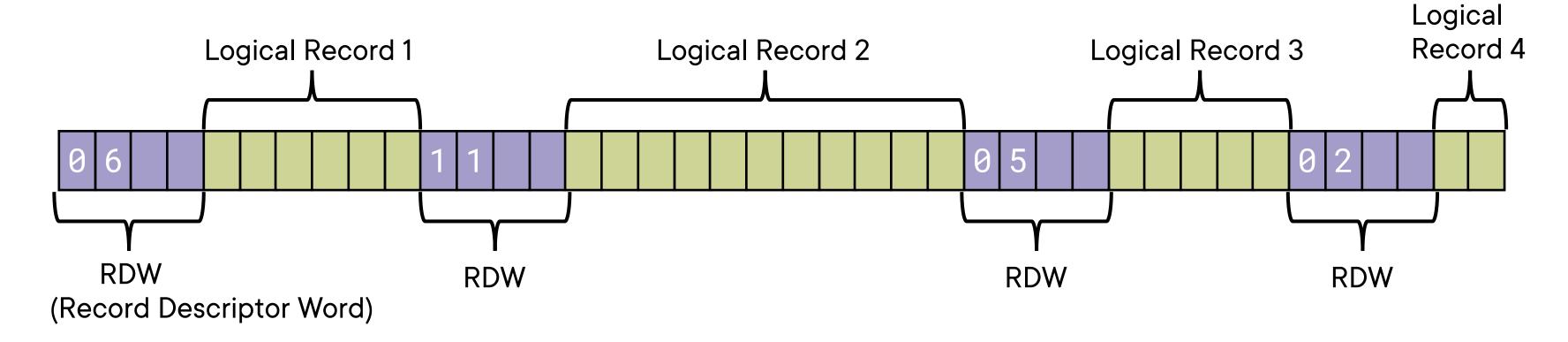
DD RECFM=FB, LRECL=10, BLKSIZE=20



Rule: Block size must be an even multiple of logical record length

## Variable-length Records, Unblocked

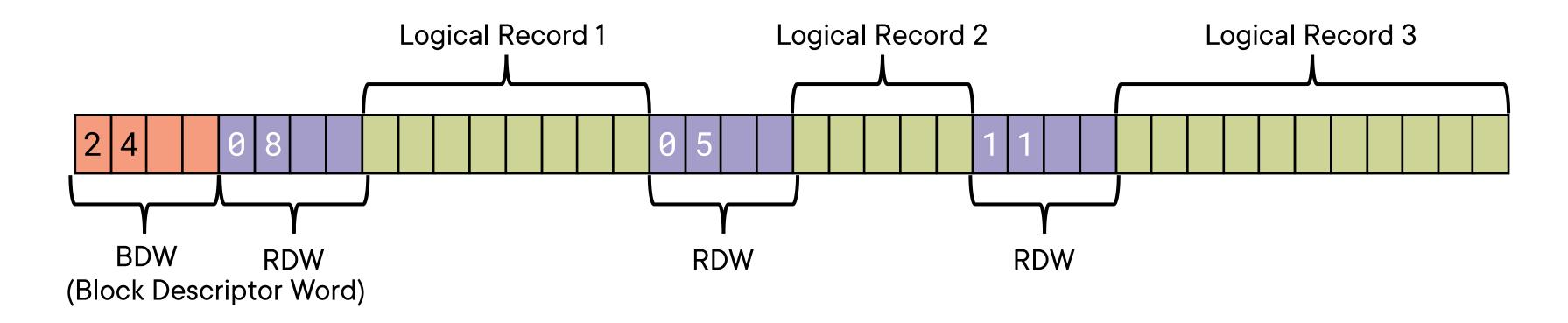
DD RECFM=V, LRECL=16, BLKSIZE=20



Rule: Block size must be at least the max logical record length + 4

## Variable-length Records, Blocked

DD RECFM=VB, LRECL=8, BLKSIZE=20



Rule: Block size must be at least (average logical record length x number of logical records per block) + 4

```
//DD1 DD DSN=MY.NEW.DATA.SET,
// DISP=(NEW,CATLG,DELETE),
// SPACE=(CYL,30),
// DSORG=PS,
// RECFM=F,
// LRECL=1024,
// BLKSIZE=1024
```

- ▼ Fixed-length records, unblocked
- Logical record length and block size are the same

```
//DD1 DD DSN=MY.NEW.DATA.SET,
// DISP=(NEW,CATLG,DELETE),
// SPACE=(CYL,30),
// DSORG=PS,
// RECFM=FB,
// LRECL=1024,
// BLKSIZE=8192
```

- **◄ Fixed-length records, blocked**
- Block size is an even multiple of the logical record length

```
//DD1 DD DSN=MY.NEW.DATA.SET,
// DISP=(NEW,CATLG,DELETE),
// SPACE=(CYL,30),
// DSORG=PS,
// RECFM=V,
// LRECL=1024,
// BLKSIZE=8196
```

- **◄ Variable-length records, unblocked**
- Block size is 4 larger than the maximum logical record length to leave room for the Record Descriptor Word (RDW)

```
//DD1 DD DSN=MY.NEW.DATA.SET,
// DISP=(NEW,CATLG,DELETE),
// SPACE=(CYL,30),
// DSORG=PS,
// RECFM=VB,
// LRECL=1024,
// BLKSIZE=8200
```

- Variable-length records, blocked
- Block size is 4 larger than the average logical record length, including RDWs, to leave room for the Block Descriptor Word (BDW)

# DD Coding for Generation Data Groups

# Concept: Generation Data Group

DATA.SET.NAME(+1)	Create new generation
DATA.SET.NAME(0) DATA.SET.NAME(-1) DATA.SET.NAME(-2)	Current generation Previous generation 2 generations ago
DATA.SET.NAME(-9999)	Oldest saved generation

### Supported Data Set Organization for GDGs



QSAM, BSAM, BDAM, PDS, PDSE



### GDG: Absolute Generation and Version

Relative	Generation	Number
NGIALIVE	<b>Generation</b>	NULLING

#### Absolute Generation & Version Number

DATA.SET.NAME(0) DATA.SET.NAME(-1) DATA.SET.NAME(-2) DATA.SET.NAME(-3) DATA.SET.NAME(-4) DATA.SET.NAME(-4) DATA.SET.NAME(-5) DATA.SET.NAME(-6) DATA.SET.NAME(-7)	DATA.SET.NAME.G0820V00 DATA.SET.NAME.G0819V00 DATA.SET.NAME.G0818V00 DATA.SET.NAME.G0817V02 DATA.SET.NAME.G0816V00 DATA.SET.NAME.G0815V01 DATA.SET.NAME.G0814V00 DATA.SET.NAME.G0813V00
DATA.SET.NAME(-8)	DATA.SET.NAME.G0812V00

# GDG: Wrap Flag and Sequencing

Absolute Generation & Version Number		Wrap Flag	Effective Number	Sequence
DATA.SET.NAME.G0003V00	(0)	1	10003	1
DATA.SET.NAME.G0002V00	(-1)	1	10002	2
DATA.SET.NAME.G0001V00	(-2)	0	10001	3
DATA.SET.NAME.G999V00	(-3)	0	9999	4
DATA.SET.NAME.G9998V00	(-4)	0	9998	5
DATA.SET.NAME.G9997V00	(-5)	0	9997	6

### GDG: Replace GDS with New Version

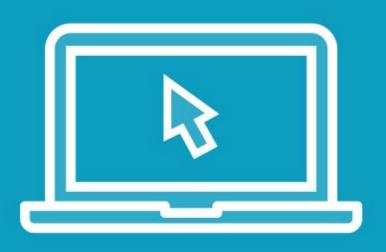
#### Before updating DATA.SET.NAME(-2)

```
DATA.SET.NAME(0)
DATA.SET.NAME.G0820V00
DATA.SET.NAME(-1)
DATA.SET.NAME.G0819V00
DATA.SET.NAME(-2)
DATA.SET.NAME.G0818V00
DATA.SET.NAME(-3)
```

#### After updating DATA.SET.NAME(-2)

```
DATA.SET.NAME(0)
DATA.SET.NAME.G0820V00
DATA.SET.NAME(-1)
DATA.SET.NAME.G0819V00
DATA.SET.NAME(-2)
DATA.SET.NAME.G0818V01
DATA.SET.NAME(-3)
DATA.SET.NAME.G0817V02
```

### Demo



#### Define a Generation Data Group and create Generation Data Sets in the group

- Use IDCAMS to define the GDG
- 2. Use IEFBR14 to create a model DCB
- 3. Specify generation (+1) on DD statements in job steps that create new Generation Data Sets

```
//DD1 DD DSN=D.S.N.G0004V00,
// DISP=(,CATLG),DSORG=PS,SPACE=(TRK,1)
```

■ Nothing stops you from cataloguing a GDS as you would any other data set. Don't do this. It defeats the GDG logic.

```
//DD1 DD DSN=DATA.SET.NAME(+1),
// DISP=(,CATLG),DSORG=PS,SPACE=(TRK,1)
```

```
//DD1 DD DSN=DATA.SET.NAME(+1),
// DISP=(,CATLG),REFDD=ddname
```

◆ Catalog a new GDS by referring to another DD
 statement that created a similar data set

```
//DD1 DD DSN=DATA.SET.NAME(+1),
// DISP=(,CATLG),DATACLAS=name
```

### Three Ways To Create a New Generation

IEFBR14
Utility

IEBGENER
Utility

User-written Program

### GDG: Considerations for Deleting a GDS

Before deleting DATA.SET.NAME(-2)

0819V00
0818V00
0817V02
0816V00
(

After deleting DATA.SET.NAME(-2)

```
DATA.SET.NAME(0)
DATA.SET.NAME(-1)
DATA.SET.NAME.G0820V00
DATA.SET.NAME(-1)
DATA.SET.NAME.G0819V00
DATA.SET.NAME(-2? 3?)
DATA.SET.NAME.G0817V02
DATA.SET.NAME.G0816V00
```

# DD Coding for Partitioned Data Sets

### Overview



- PDS vs. PDSE formats and usage
- Allocating space
- Library for data members
- Library for program objects
- IEBCOPY utility

### Two Kinds of PDSs

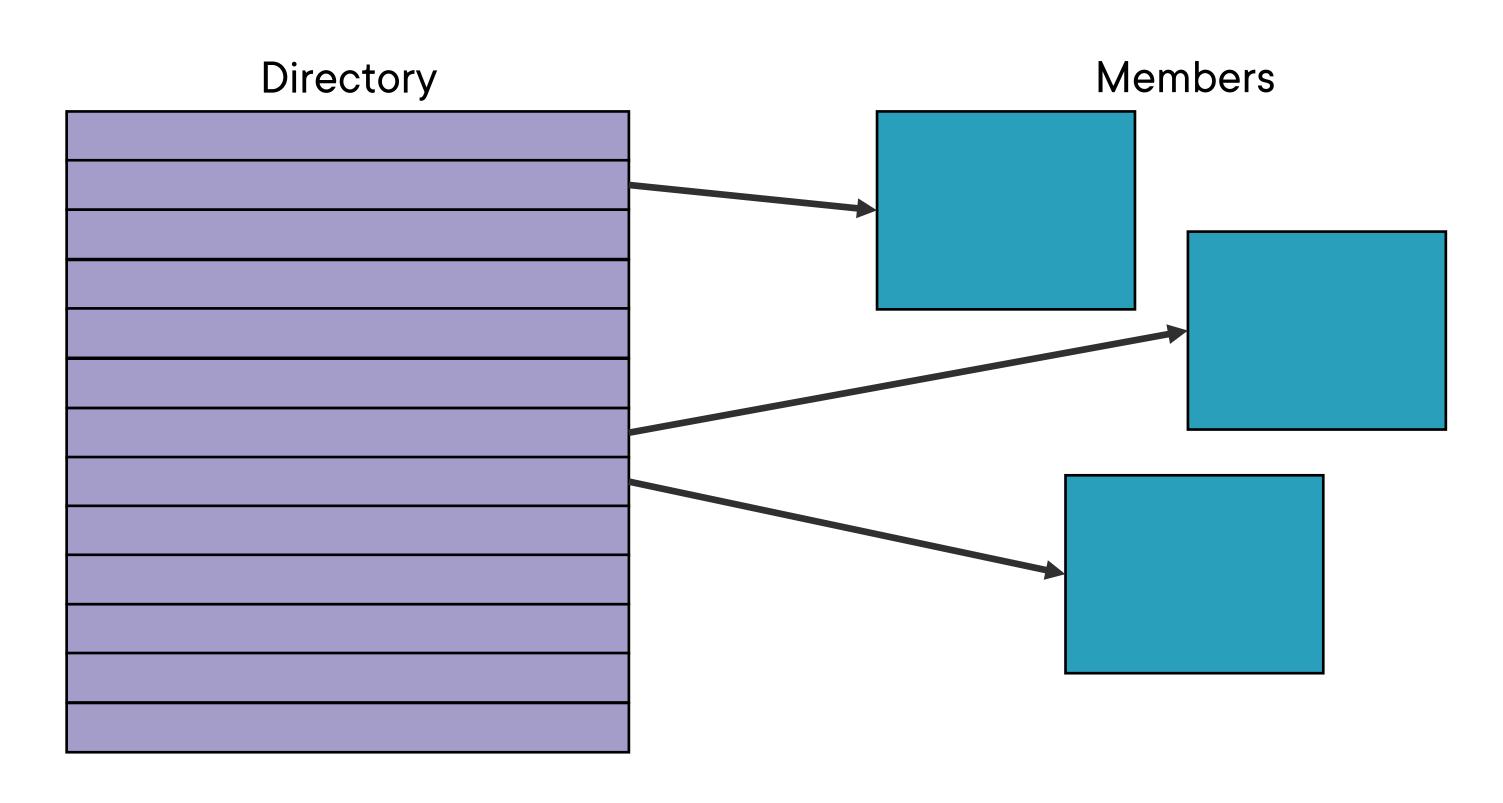
#### **PDS**

The original implementation, now supported for backward compatibility

#### **PDSE**

The current standard with many improvements and enhanced capabilities

# PDS Directory and Members



### PDSE Contents

### **Data Members**

Text, program source code, configuration files, etc.

### Program Objects

Executable binaries

```
//DD1 DD DSN=MY.NEW.PDS,
         DISP=(NEW, CATLG, DELETE),
//
         DSNTYPE=LIBRARY,
//
//
//DD1 DD DSN=MY.NEW.PDS,
         DISP=(NEW, CATLG, DELETE),
//
//
         DSNTYPE=PDS,
```

**◄** Specifies PDSE

**◄** Specifies PDS (legacy)

```
//DD1 DD DSN=MY.NEW.PDS,
         DISP=(NEW, CATLG, DELETE),
//
//
         DSNTYPE=LIBRARY,
         DCB=(RECFM=FB, LRECL=100),
//
//
//DD1 DD DSN=MY.NEW.PDS,
         DISP=(NEW, CATLG, DELETE),
//
//
         DSNTYPE=LIBRARY,
         DCB=(RECFM=U)
//
```

■ PDSE containing data members

**◄ PDSE** containing program objects

```
//DD1 DD DSN=MY.NEW.PDS,
         DISP=(NEW, CATLG, DELETE),
//
//
         DSNTYPE=LIBRARY,
         SPACE=(CYL, (10, 10, 10)),
//
//
//DD1 DD DSN=MY.NEW.PDS,
         DISP=(NEW, CATLG, DELETE),
//
         DSNTYPE=LIBRARY,
         SPACE=(300, (5000, 100)),
```

**◄** Space allocation in cylinders

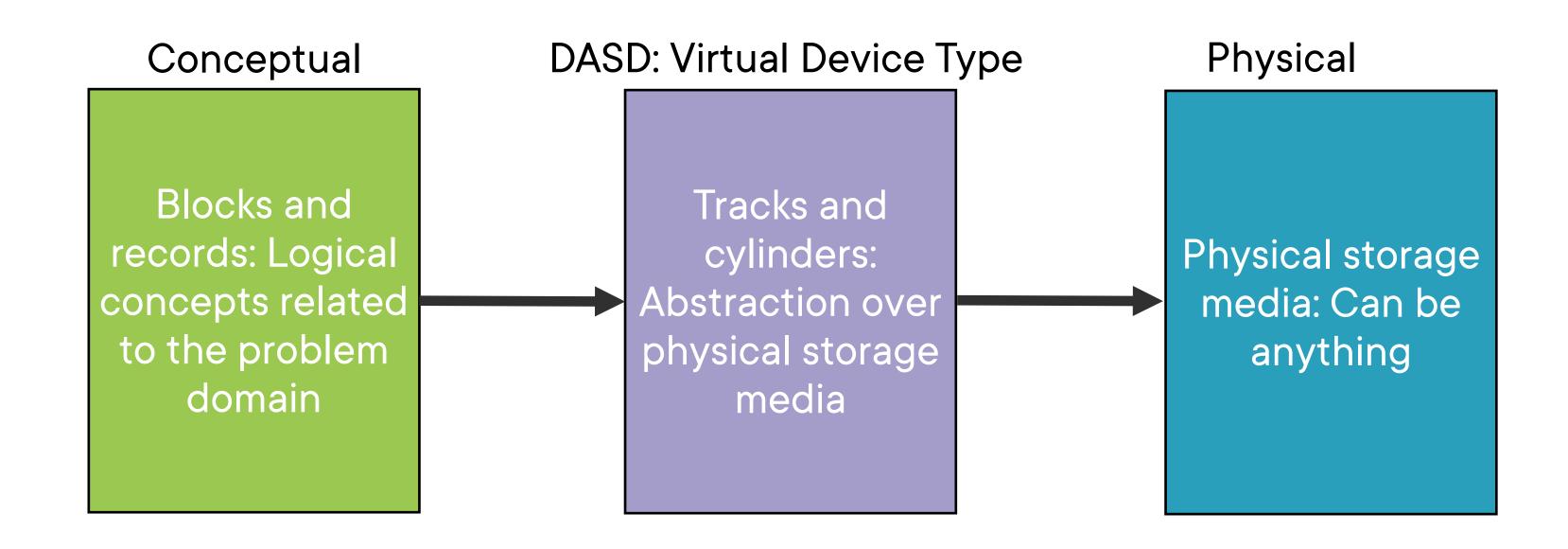
✓ Space allocation in blocks
 300 = average block size
 5000 = number of blocks in primary extent
 100 = number of blocks in secondary extent

```
//DD1 DD DSN=MY.NEW.PDS,
         DISP=(NEW, CATLG, DELETE),
//
//
         DSNTYPE=LIBRARY,
         SPACE=(80,(20,2)),
//
//
         AVGREC=K,
//DD1 DD DSN=MY.NEW.PDS,
         DISP=(NEW, CATLG, DELETE),
//
         DSNTYPE=LIBRARY,
         SPACE=(200, (8,1)),
         AVGREC=M,
```

Space allocation in records
80 = average record length in bytes
AVGREC=K means the values in parentheses
are in terms of kilobytes
20 = primary allocation 20K
2 = secondary allocation 2K

✓ Space allocation in records
 200 = average record length in bytes
 AVGREC=M means the values in parentheses
 are in terms of megabytes
 8 = primary allocation 8M
 1 = secondary allocaiton 1M

# Blocks & Records Related to Tracks & Cylinders



```
//DD1 DD DSN=MY.NEW.PDS,
         DISP=(NEW, CATLG, DELETE),
//
//
         DSNTYPE=LIBRARY,
         SPACE=(CYL, (20,4)),
//
         DCB=(DSORG=PO,...),
//
//DD1 DD DSN=MY.NEW.PDS,
         DISP=(NEW, CATLG, DELETE),
//
         DSNTYPE=LIBRARY,
         DATACLAS=name,
```

**◄ Let the system calculate directory space** 

**◄ Reference a data class** 

**◄ Using Access Method Services to create PDSE** 

# DD Coding for VSAM Data Sets

### Overview



- What is VSAM?
- VSAM data set formats
- VSAM access modes
- IDCAMS utility

# VSAM Functional Components

### Catalog Management

Works in concert with ICF to manage VSAM catalogs

### Record Management

Works in concert with SMS to manage space allocation for VSAM data sets

# VSAM Data Set Types

Key Sequenced Data Set (KSDS)
Records stored in sequence based on a logical record key

Entry Sequenced Data Set (ESDS)
Records stored in the order in
which they are added

Relative Record Data Set (RRDS)
Records are numbered and stored in the order of their numbers

Linear Data Set (LDS)
Functions as a byte stream

### Three Access Modes

Sequential

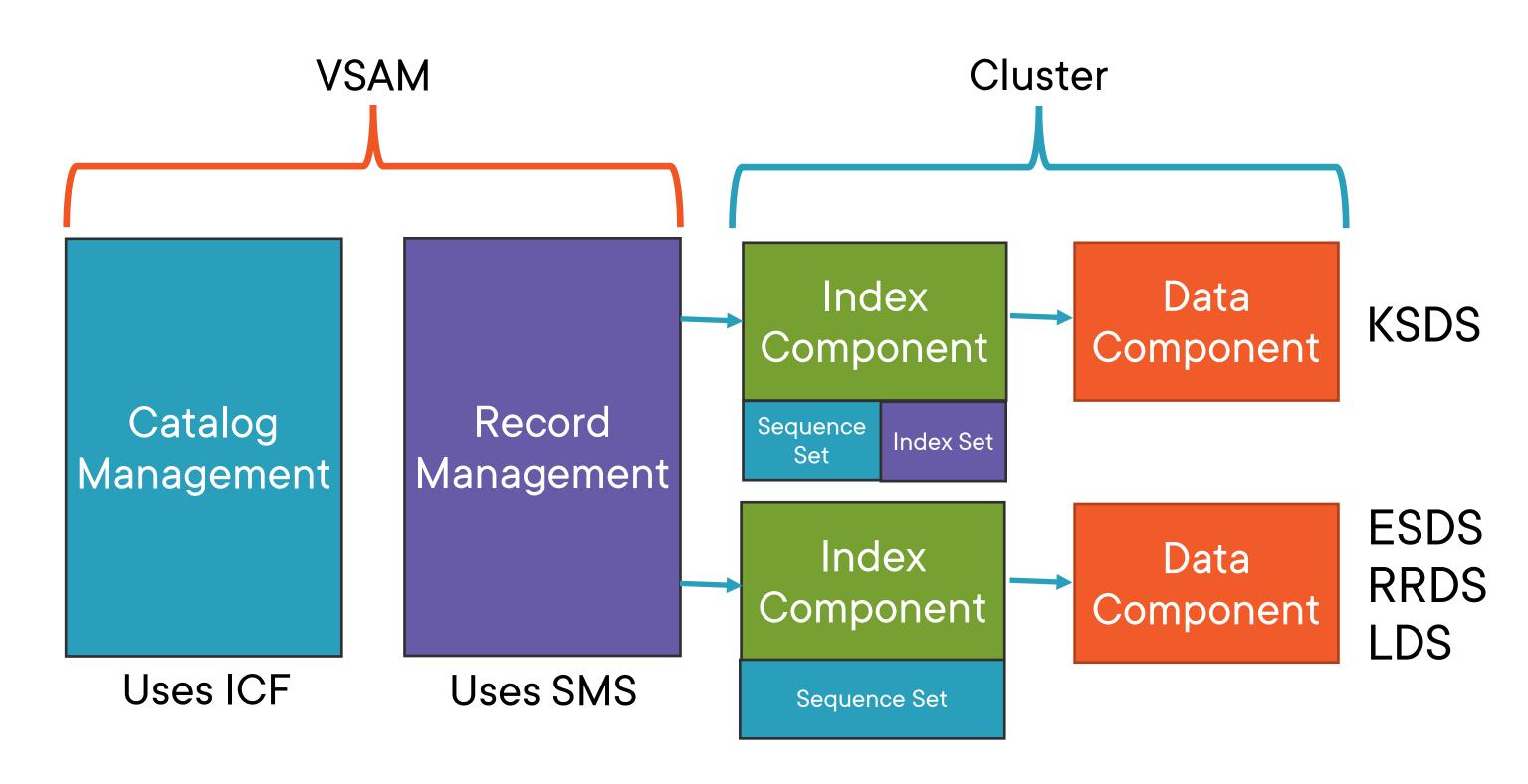
Direct

Skip-Sequential

# VSAM Data Set Types and Access Modes

Type	Record Format	Sequential	Direct	Skip-sequential
KSDS	Fixed, Variable	Yes	Yes	Yes
ESDS	Fixed, Variable	Yes	No	No
RRDS	Fixed	Yes	Yes	Yes
LDS	Byte Stream	Yes	No	No

# VSAM Components and Clusters



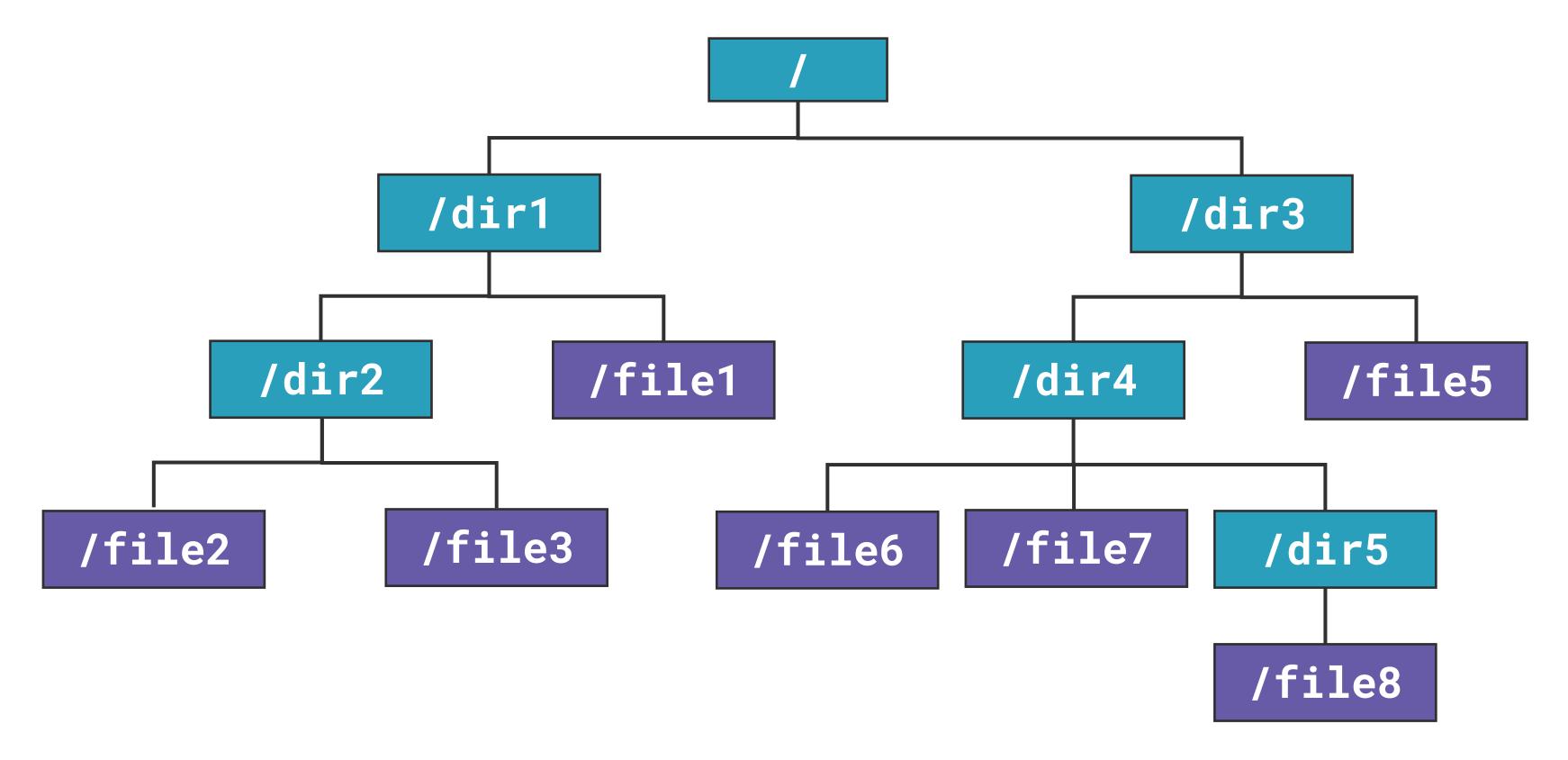
# DD Coding for HFS Files

### Overview



- What is HFS?
- Directories, files, paths
- DD parameters for HFS files

# Hierarchical File Systems on USS



```
//DD1 DD DSN=MY.NEW.DATA.SET,
// DISP=(NEW,CATLG,DELETE),
// ...
```

```
■ Instead of a DSNAME parameter specifying a data set name...
```

```
//DD1 DD PATH='/u/dir1/dir2/filename',
// DISP=(NEW, KEEP, DELETE),
//
```

```
//DD1 DD PATH='/u/dir1/dir2/filename',
// PATHDISP(KEEP, DELETE),
//
//DD1 DD PATH='/u/dir1/dir2/filename',
// PATHDISP(KEEP, DELETE),
// FILEDATA=TEXT,
//
```

```
◄ Code PATHDISP instead of DISP
```

◄ FILEDATA tells the system what kind of file this is. Possible values are:

BINARY – binary file

TEXT – text file

RECORD – BSAM, QSAM, VSAM, or BPAM

```
//DD1 DD PATH='/u/dir1/dir2/filename',
// PATHDISP(KEEP, DELETE),
// FILEDATA=TEXT,
// PATHOPTS=(OCREAT, ORDWR),
//
```

▼ PATHOPTS tells the system how to open the file. To create the file in this example, we specify OCREAT (create) and ORDWR (open for reading and writing).

(Note: Don't create these data set types this way)

# DD PATHOPTS Subparameters

### Access Group (specify only one)

```
ORDONLY Program opens file for reading OWRONLY Program opens file for writing ORDWR Program opens file for reading and writing
```

### Status Group (can specify multiples)

```
OAPPEND Program will append to the end of the file
OCREAT File will be created — also see OEXCL
OEXCL System will create the file if it doesn't exist
System will fail the step if the file exists
OEXCL only effective if OCREAT is specified
```

See z/OS MVS JCL Reference for details of more subparameters

```
//DD1 DD PATH='/u/dir1/dir2/filename',
// PATHDISP(KEEP, DELETE),
// FILEDATA=TEXT,
// PATHOPTS=(OCREAT, ORDWR),
// PATHMODE=(SIRUSR, SIWUSR,
// SIRGRP, SIWGRP, SIROTH)
// ...
```

 The PATHMODE parameter sets the privileges for the file − same as the file mode on Unix and Linux.

## DD PATHMODE Parameter

### PATHMODE Value

### SIRUSR

SIWUSR

SIXUSR

SIRWXU

**SIRGRP** 

SIWGRP

SIXGRP

SIRWXG

SIROTH

SIWOTH

SIXOTH

SIRWXO

### Unix/Linux Equivalent

```
400 r-- ---
```

$$020 \quad --- \quad -w - \quad ---$$

$$010 \quad --- \quad -x \quad --$$

# DD PATHMODE Examples

PATHMODE=(SIRUSR, SIWUSR, SIRGRP, SIXGRP, SIROTH, SIXOTH)

655 rw- r-x r-x

PATHMODE=(SIRWXU, SIXGRP, SIXOTH)

711 rwx --x --x

PATHMODE=(SIRUSR, SIWUSR, SIRGRP, SIWGRP, SIROTH, SIWOTH)

666 rw- rw- rw-

## Connecting Data Sets with Job Steps

## Separating File Names & Data Set Names

## Meaning

File name inside a program denotes a domain concept; z/OS DSN is cryptic

## **Flexibility**

Ability to process different data sets with same program

### Portability

Ability to run the program in different z/OS execution contexts

# Connecting File Names & Data Set Names

System knows...

Data Set Name

DD Name

Program knows...

## The DDNAME Is the Connector

```
...
//INPUT DD DSN=input.data.set,DISP=SHR
...
```

#### Generic

## Assembler

```
. . . . //INPUT DD DSN=input.data.set,DISP=SHR . . . .
```

### Assembler

## COBOL

```
. . . . //INPUT DD DSN=input.data.set,DISP=SHR . . . .
```

#### COBOL

```
FILE-CONTROL.

SELECT PEOPLE-TO-GREET

ASSIGN TO 'INPUT'

OPEN INPUT PEOPLE-TO-GREET

. . .
```

# PL/I

```
. . . . //INPUT DD DSN=input.data.set,DISP=SHR . . . .
```

### PL/I

```
DCL NAMES FILE INPUT RECORD SEQUENTIAL BUFFERED;
OPEN FILE (NAMES) TITLE ('INPUT');
. . .
```

## Rexx

```
. . .
//INPUT DD DSN=input.data.set,DISP=SHR
. . .
```

### Rexx

```
"EXECIO * DISKR INPUT (FINIS STEM GREETS."
DO I = 1 TO GREETS.0
. . .
```

## Java

```
...
//INPUT DD DSN=input.data.set,DISP=SHR
...
```

#### Java

# C/C++

```
. . . . //INPUT DD DSN=input.data.set,DISP=SHR . . . .
```

```
C/C++

file *infile;
infile = fopen("dd:input","r",blksize=800,lrecl=80)
. . . .
```

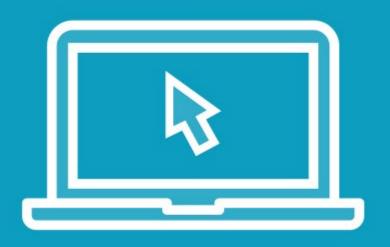
# JOBLIB, STEPLIB, and DD Concatenation

### Overview



- DD concatenation
- System search order
- JOBLIB and STEPLIB

### Demo



### **DD** Concatenation

- Create some test data sets to play with
- Write JCL and run jobs to illustrate the effect of DD concatenation

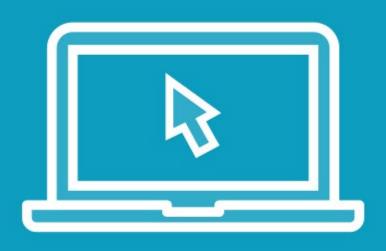
```
//jobname JOB acct, 'name', etc...
//JOBLIB DD DSN=a.b.c1, DISP=SHR
// DD DSN=a.b.c2, DISP=SHR
//STEP1 EXEC PGM=PROG1
    . . .
//STEP2 EXEC PGM=PROG2
    . . .
//STEP3 EXEC PGM=PROG3
```

■ JOBLIB will be searched in each step a.b.c1 first a.b.c2 second

```
//jobname JOB acct, 'name', etc...
//JOBLIB DD DSN=a.b.c1, DISP=SHR
// DD DSN=a.b.c2, DISP=SHR
//STEP1 EXEC PGM=PROG1
//STEPLIB DD DSN=a.b.c3, DISP=SHR
//STEP2 EXEC PGM=PROG2
//STEP3
       EXEC PGM=PROG3
//STEPLIB DD DSN=a.b.c2, DISP=SHR
         DD DSN=a.b.c1, DISP=SHR
```

- **◄ JOBLIB** will be searched in steps that don't have a STEPLIB
- The system will look for program PROG1 in library a.b.c3
- The system will look for program PROG2 in the
  JOBLIB concatenation
- The system will look for program PROG3 in library a.b.c2 first, then a.b.c1

### Demo



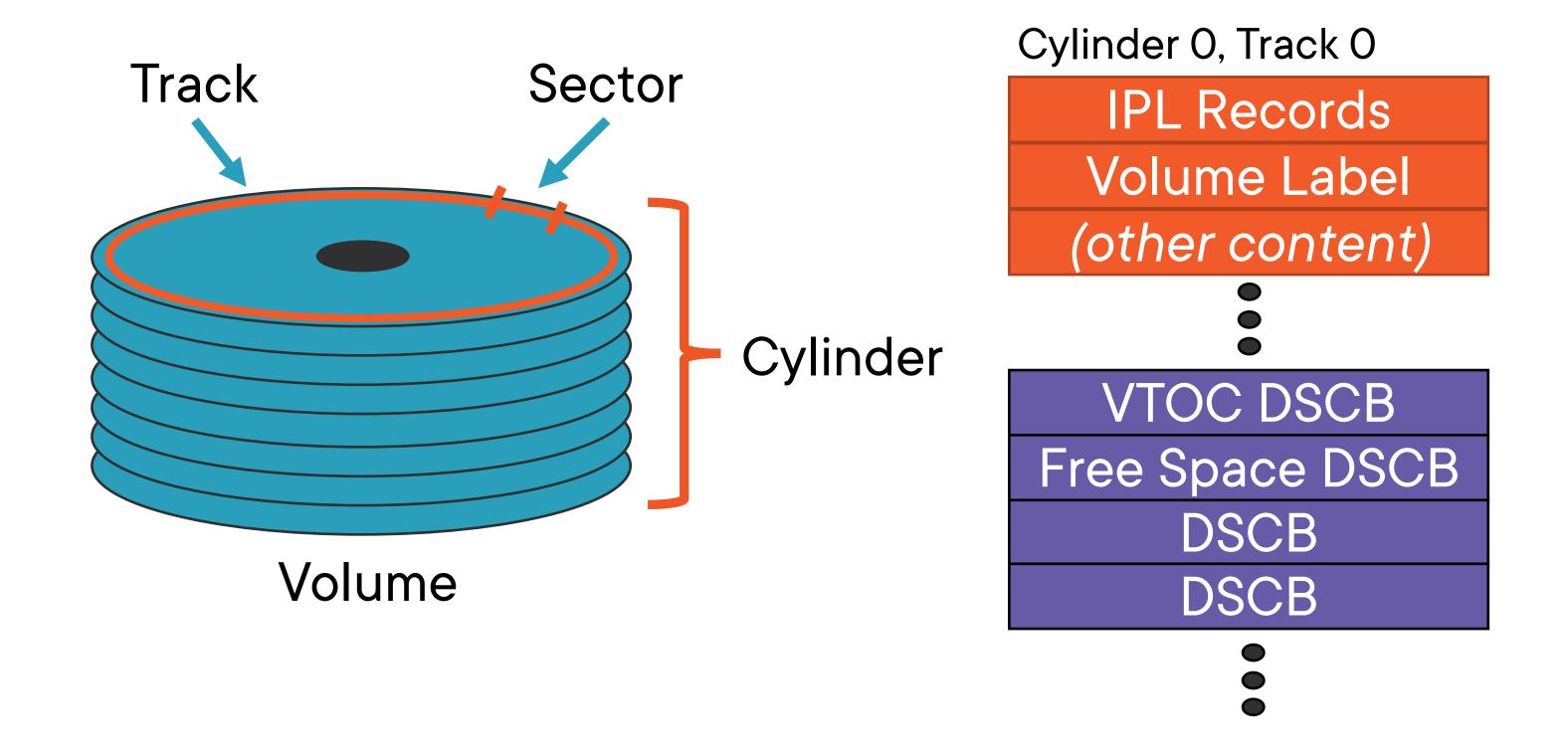
### JOBLIB and STEPLIB

- Define two libraries
- Create a modified version of a Hello, World! program
- Show the effects of different JOBLIB and STEPLIB definitions

# Module Summary

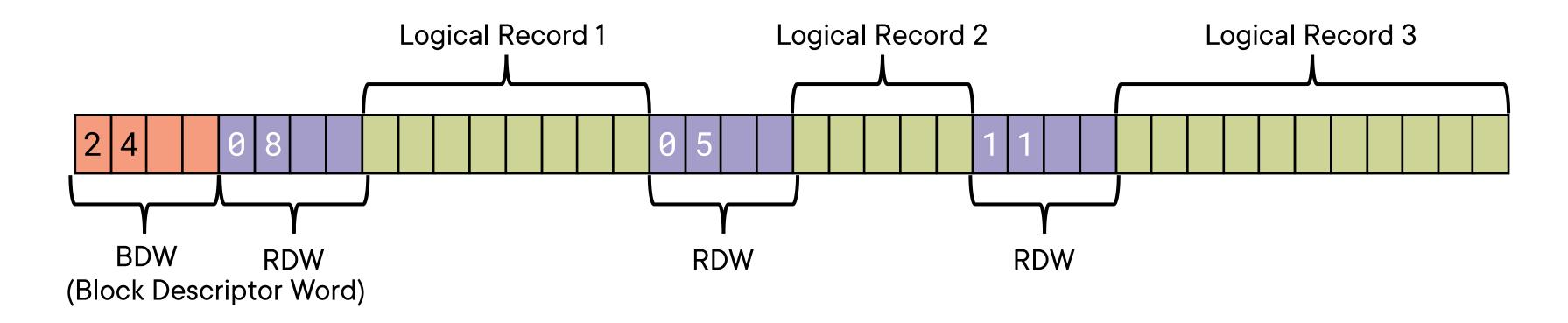


## DASD - Direct Access Storage Device



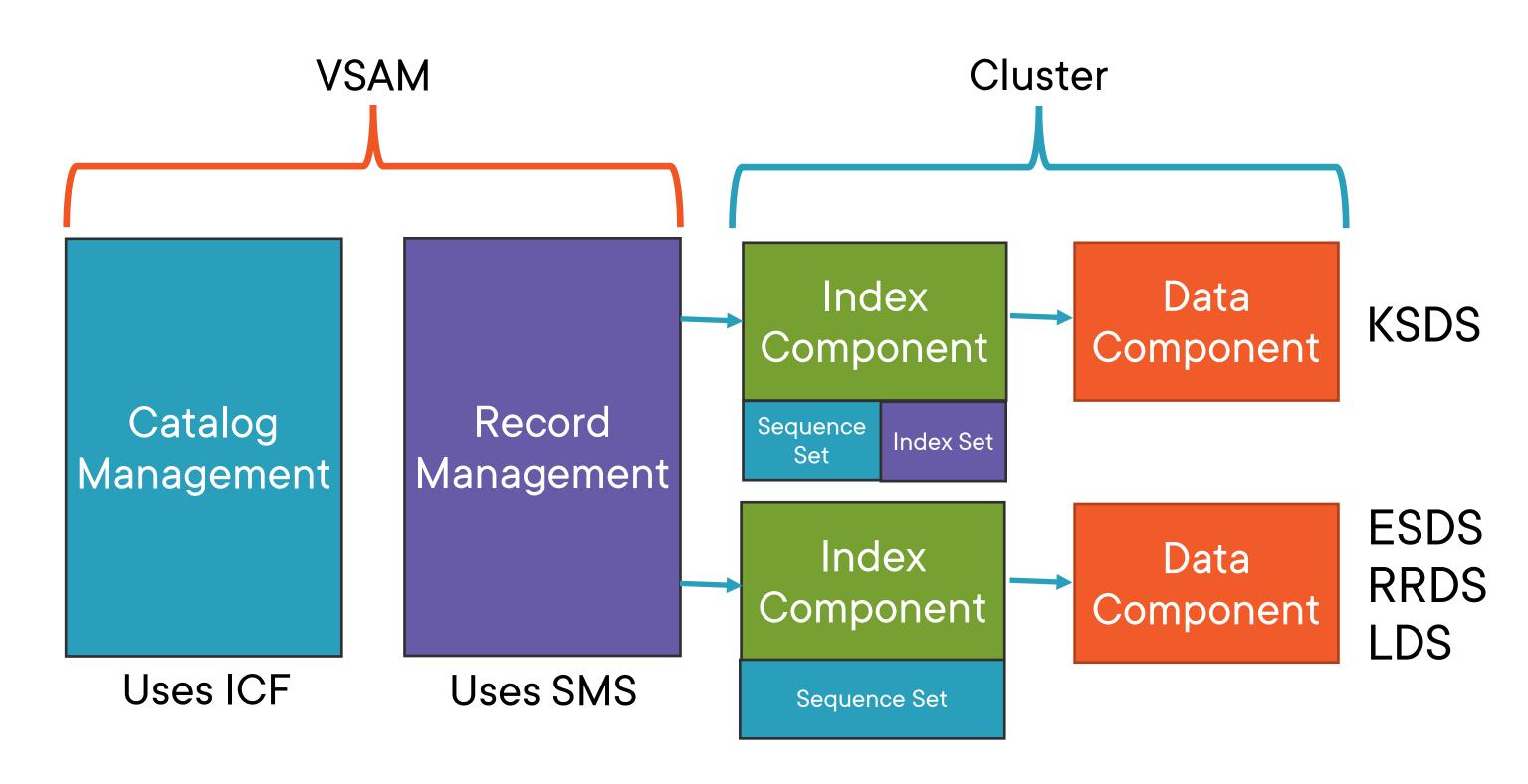
## Variable-length Records, Blocked

DD RECFM=VB, LRECL=8, BLKSIZE=20



Rule: Block size must be at least (average logical record length x number of logical records per block) + 4

# VSAM Components and Clusters



## Most Frequently-used Data Set Types

### **QSAM**

Queued Sequential Access Method

### GDG

Generation Data Group (GDG)

### **BPAM**

Basic Partitioned Access Method

### **VSAM**

Virtual Sequential Access Method

### **HFS File**

POSIX file (Unix System Services)

## COBOL

```
DD DSN=input.data.set,DISP=SHR
COBOL
 FILE-CONTROL.
     SELECT PEOPLE-TO-GREET
     ASSIGN TO 'INPUT'
     OPEN INPUT PEOPLE-TO-GREET
```

```
//jobname JOB acct, 'name', etc...
//JOBLIB DD DSN=a.b.c1, DISP=SHR
// DD DSN=a.b.c2, DISP=SHR
//STEP1 EXEC PGM=PROG1
//STEPLIB DD DSN=a.b.c3, DISP=SHR
//STEP2 EXEC PGM=PROG2
//STEP3
       EXEC PGM=PROG3
//STEPLIB DD DSN=a.b.c2, DISP=SHR
         DD DSN=a.b.c1, DISP=SHR
```

- **◄ JOBLIB** will be searched in steps that don't have a STEPLIB
- The system will look for program PROG1 in library a.b.c3
- The system will look for program PROG2 in the
  JOBLIB concatenation
- The system will look for program PROG3 in library a.b.c2 first, then a.b.c1

Up Next:

Examining How the System Processes JCL