COMP 3400 Mainframe Administration[1]

Christian Grothoff
christian@grothoff.org

http://grothoff.org/christian/

[1]These slides are based in part on materials provided by IBM's Academic Initiative.

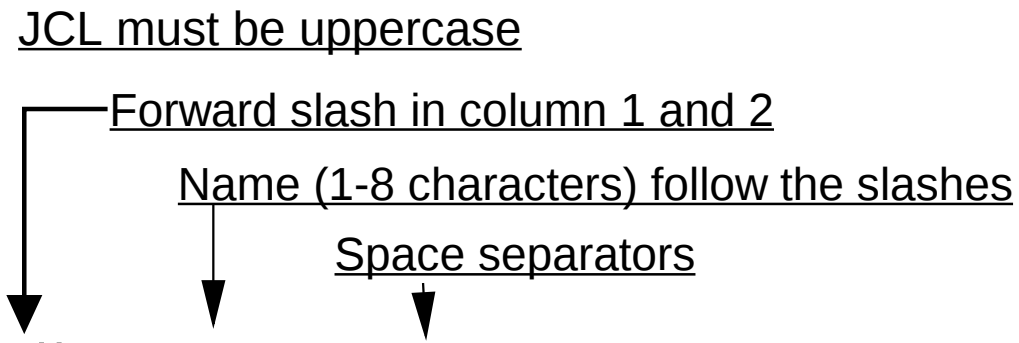
Today

- Job Control Language (JCL)
- System Display and Search Facility (SDSF)

JCL

- JCL tells the system what program to execute together with a description of program inputs and outputs.
- There are three basic JCL statements:
  - JOB
  - EXEC
  - DD

Basic JCL syntax




```
//JOBNAME   JOB
//STEPNAME  EXEC
//DDNAME    DD
//* comment - upper or lower case
/*  ....end of JCL stream
```

Continuation

JCL was initially written on punch cards:

- At most 80 characters per line, colums 73 through 80 are reserved, 72 should be a space or comma
- *Continuation* allows a JCL statement to span multiple lines
- The n-th line must end with a ","; then the $n+1$ line starts with "//" plus one or more spaces (usually as many as the NAME has)

Example

```
//MYJOB      JOB 1
//MYSORT     EXEC PGM=SORT
//SORTIN     DD DISP=SHR,DSN=IBMUSER.AREA.CODES
//SORTOUT    DD SYSOUT=*
//SYSOUT     DD SYSOUT=*
//SYSIN      DD *
  SORT FIELDS=(1,3,CH,A)
/*
```

Names in the Example

MYJOB Job name

MYSORT Step name

SORTIN Name for program input

SORTOUT Name for program output

SYSIN Name for system input (control statements)

SYSOUT Name for system output messages

The JOB Statement

In addition to giving the JOB a name, the JOB statement can be used to specify:

- Accounting information (who should be charged for time) – this is usually installation-specific
- Execution classes (permissions, username)

Syntax:

```
//JOBNAME JOB ACCINFO[,USERNAME](,KEY=VALUE)*
```

JOB Parameters

REGION= Requests specific memory resources to be allocated

NOTIFY= Send notification to specified user

USER= Assume authority of the given user

TYPRUN= Delay job start

CLASS= Execute on a particular (JES) input queue

MSGCLASS= Direct output to a particular output queue

MSGLEVEL= Control number of system messages

Parameters that are not specified result in the use of a `default` value.

Example

```
//MYJOB   JOB 19,MYUSERNAME,NOTIFY=&SYSUID,
//        MSGCLASS=T,MSGLEVEL=(1,1),CLASS=A,
//        REGION=16M
```

The End of a Job

The "scope" of a job statement ends at:

- the next job statement
- a null statement ("// ")
- the end of file indicator (/*)

The EXEC statement

A JCL script can consist many EXEC statements; they specify:

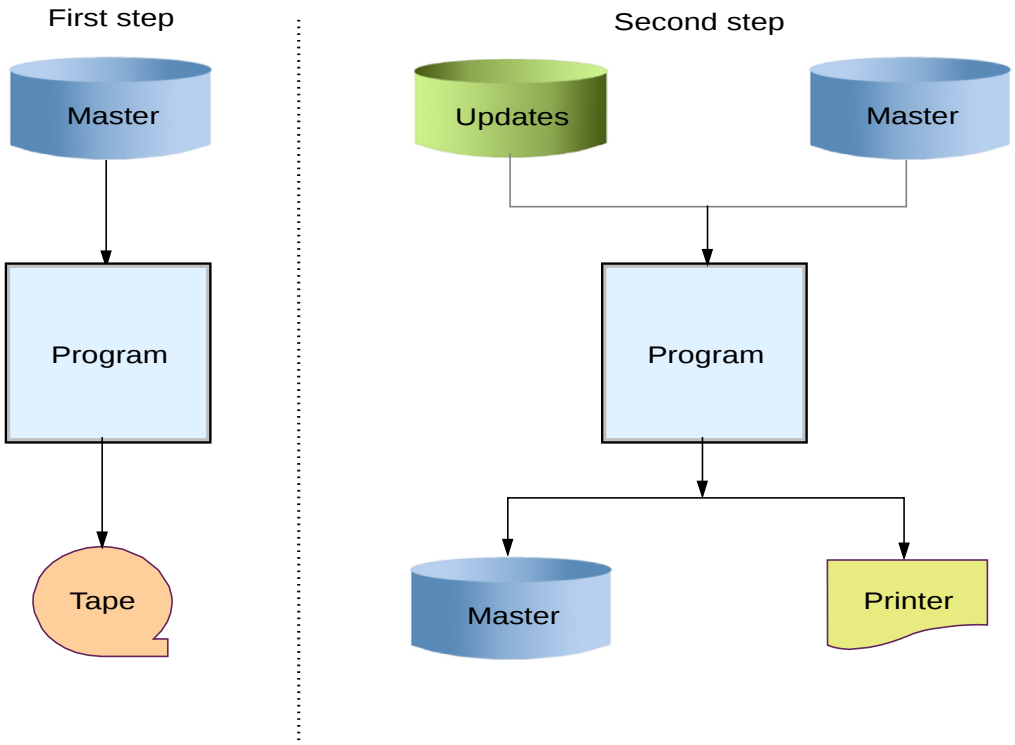
- The step name
- The program to execute
- Resource requirements for the program (region size)

Example

```
//STEP42   EXEC PGM=IEFBR14
```

Note that `PGM=IEFBR14` is a **positional** parameter and **must** come first after `EXEC` (it is not like the other `EXEC` parameters!).

Multi-Step Jobs



EXEC Parameters

PARM= Parameters passed to the program

COND= Guards (boolean logic for controlling execution)

Guards

Guards (given as `COND` parameters) to EXEC can be used to ensure that steps only execute conditionally:

- Modern JCL also has `IF THEN ELSE`, but older code uses `COND`
- Example: `COND=(0,NE)`: execute only if the previous return code was equal to zero (if condition is TRUE, skip!)

The DD Statement

Each EXEC statement can have a number of DD statements, which specify:

- The DD name (referenced in the program; remember COBOL?)
- The name of the input or output – or the input data directly
- For outputs, how to allocate space
- The disposition of the data sets (what to do with them after execution)

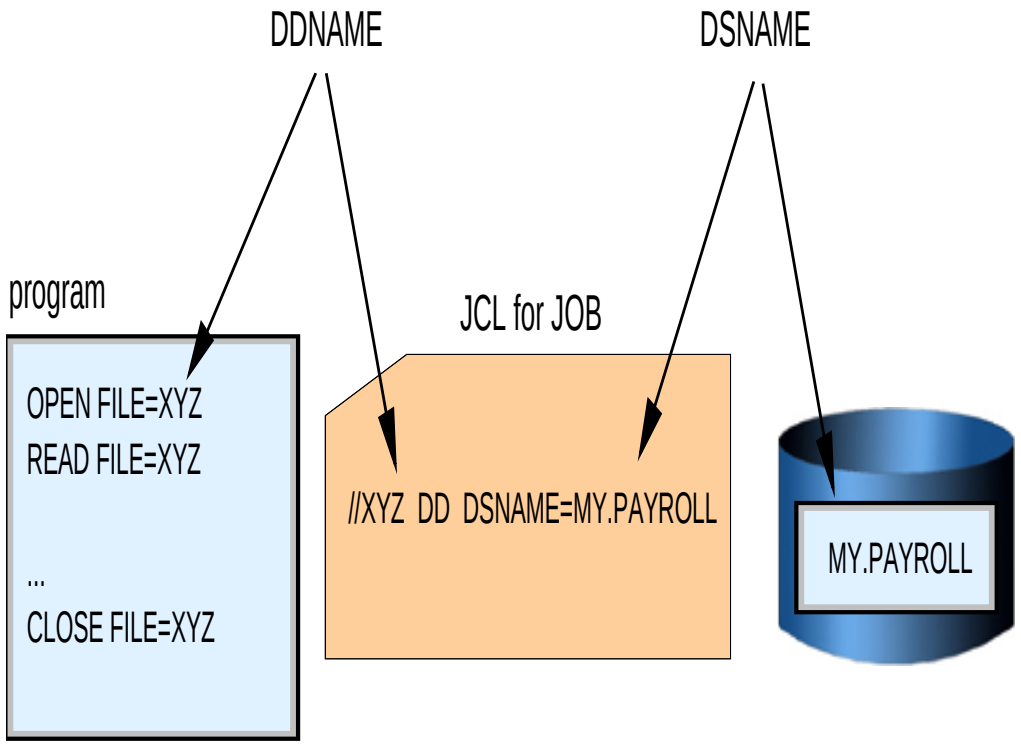
Example

```
//SOMEDD DD DSN=USERNAME.SOME.DATA.SET,
//          DISP=(NEW,CATLG,DELETE),
//          UNIT=SYSDA,
//          SPACE=(CYL,(10,10,4)),
//          LRECL=80,BLKSIZE=3120
```

Symbolic References to Files

- z/OS normally uses symbolic references to files (data sets) rather than actual file names.
- The use of symbolic references provides a naming redirection between a data set-related name used in a program and the actual data set used during execution of that program.

Symbolic References to Files



Data Set Dispositions

The `DISP` operand specifies the data set disposition:

- Three arguments, for "at step start", "at normal step end" and "at abnormal step end"
- DISP can be used to prevent unwanted concurrent access to data set

The default disposition (for normal and abend) is to leave the data set as it was before the job started.

Uses of the DISP operand

You can specify DISP operands in three ways:

- DISP=(status,normal end,abnormal end)
- DISP=(status,normal end)
- DISP=status

Possible values for "status" can be NEW (allocate, must not exist, exclusive), OLD (exclusive), SHR (shared access) or MOD (exists, exclusive, append output).

Uses of the DISP operand

For the *normal* and *abnormal end* dispositions, the following choices are possible:

- DELETE
- KEEP (does not change catalog status)
- CATLG (keep and catalog)
- UNCATLG (keep and remove from catalog)
- PASS (allow later job do specify final DISP)

Using DISP=NEW

If you allocate a data set using `DISP=NEW` you need to specify additional arguments, including:

- A data set name (`DSN=`)
- The type of device for the data set (`UNIT=`)
- If a DASD is used, the amount of space to be allocated for the primary extent (`SPACE=`)
- If a PDS is desired, the size of the directory must be specified within the `SPACE` parameter
- You can also specify the `LRECL` and `BLKSIZE`

Concatenation

Concatenation allows a single DD name to have multiple DD statements:

$\Rightarrow$ Can process multiple data sets in one step!

Example

```
//DATAIN DD DISP=OLD,DSN=MY.INPUT1
//        DD DISP=OLD,DSN=MY.INPUT2
//        DD DISP=SHR,DSN=YOUR.DATA
```

JCL Procedures

- Defined using the "PROC" statement
- End with the "PEND" statement
- "arguments" are simply strings prefixed with "&"

Example: Procedure definition

```
//MYPROC    PROC
//MYSORT    EXEC PGM=SORT
//SORTIN    DD DISP=SHR,DSN=&SORTDSN
//SORTOUT   DD SYSOUT=*
//SYSOUT    DD SYSOUT=*
//          PEND
```

Example: Using the Procedure

```
//MYJOB      JOB 1
//MYPROC     PROC
//MYSORT     EXEC PGM=SORT
//SORTIN     DD DISP=SHR,DSN=&SORTDSN
//SORTOUT    DD SYSOUT=*
//SYSOUT     DD SYSOUT=*
//           PEND
//STEP1      EXEC MYPROC,SORTDSN=IBMUSER.AREA.CODES
//SYSIN      DD *
  SORT FIELDS=(1,3,CH,A)
```

Statement Overrides

```
//MYJOB      JOB 1
//MYPROC     PROC
//MYSORT     EXEC PGM=SORT
//SORTIN     DD DISP=SHR,DSN=&SORTDSN
//SORTOUT    DD SYSOUT=*
//SYSOUT     DD SYSOUT=*
//           PEND
//STEP1      EXEC MYPROC,SORTDSN=IBMUSER.AREA.CODES
//MYSORT.SORTOUT DD DSN=IBMUSER.MYSORT.OUTPUT,UNIT=SYSDA,
//           DISP=(NEW,CATLG),SPACE=(CYL,(1,1)),
//           DCB=(LRECL=20,BLKSIZE=0,RECFM=FB,DSORG=PS)
//SYSIN      DD *
  SORT FIELDS=(1,3,CH,A)
```

Locations of Code

SYS1.LINKLIB Execution modules of the system (`/sbin/`)

SYS1.LPALIB Execution modules loaded into the LPA, common storage shared by all address spaces (example: `IEFBR14`)

SYS1.PROCLIB JCL procedures distributed with z/OS (`/usr/`)

SYS1.PARMLIB Control parameters for the system (`/etc`)

SYS1.NUCLEUS z/OS kernel (`/boot/`)

The search order is described in z/OS Basics, section 16.3.10.

Submitting your JCL Jobs

You can use:

- `SUBMIT` (ISPF editor command)
- `SUBmit 'my.jcl.data.set.name'` (ISPF shell)
- `TSO SUBmit 'my.jcl.data.set.name[(PDSNAME)]'` (ISPF command line)
- `SUBmit 'my.jcl.data.set'` (TSO command line)

Questions

?

Questions!

- Why has the advent of database systems potentially changed the need for large numbers of DD statements?
- The first positional parameter of a JOB statement is an accounting field. How important is accounting for mainframe usage? Why?

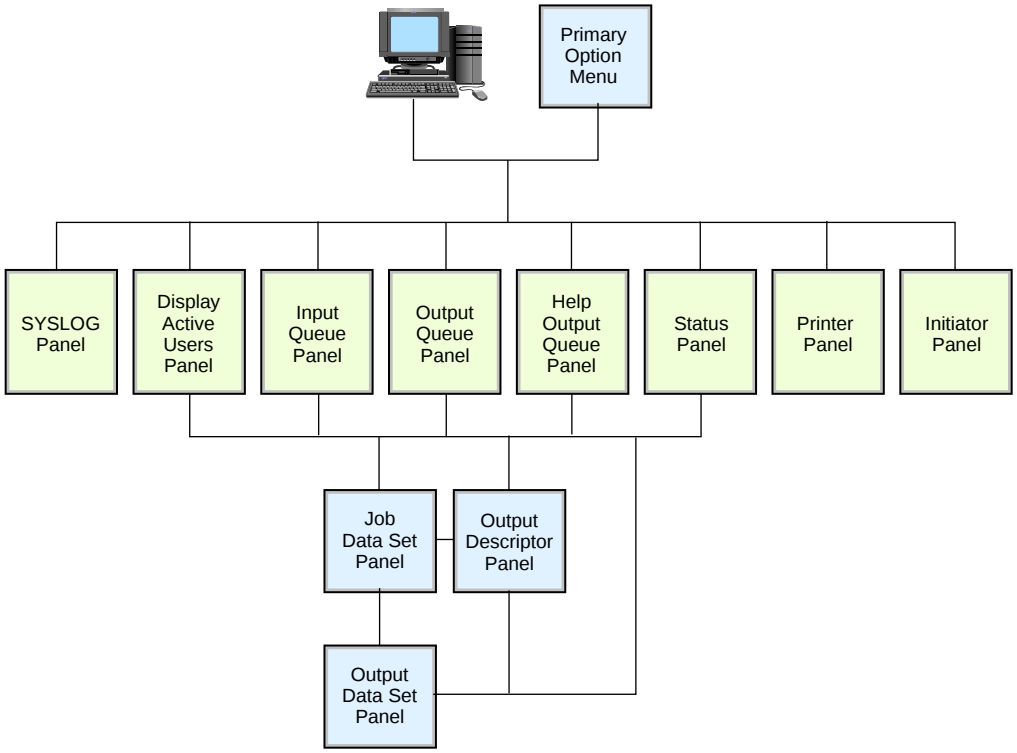
The System Display and Search Facility

SDFS allows users to control and review jobs:

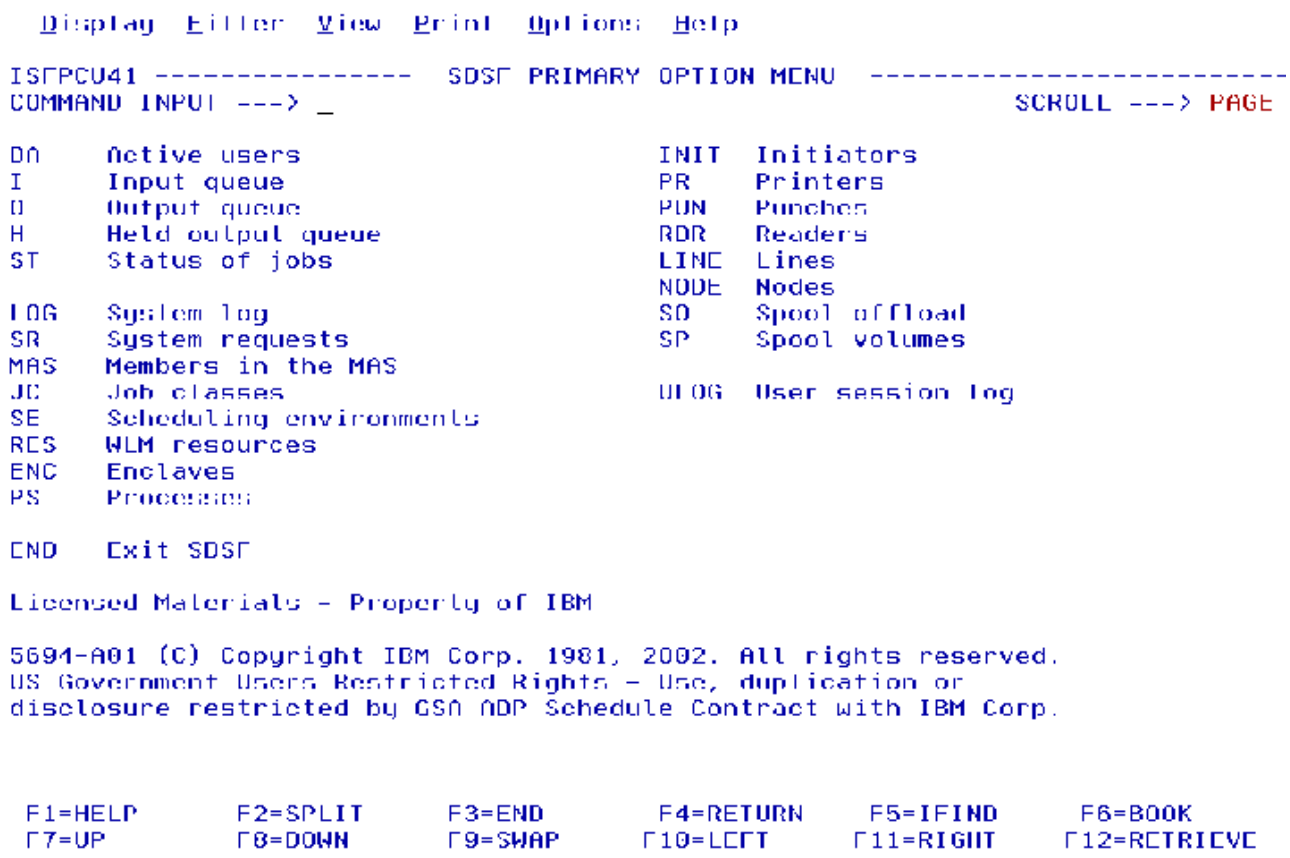
- Control jobs: hold, release, cancel, purge
- Monitor jobs while they are processing
- Display or print job output
- Control printers and initiators

SDFS can be started from the primary ISPF menu or TSO.

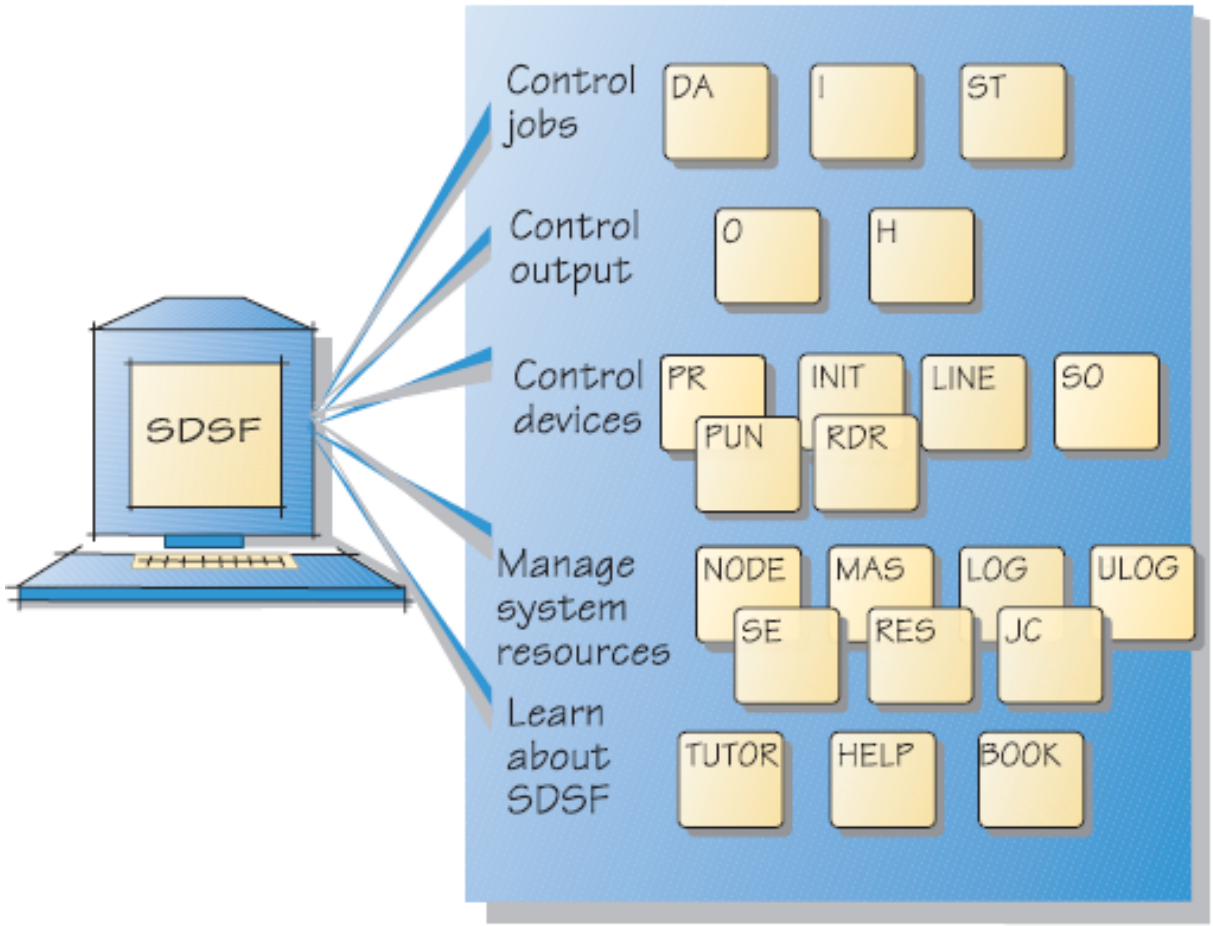
SDSF Overview



SDSF Primary Option Menu



SDSF Tasks


SDSF
Control jobs
DA
I
ST
Control output
O
H
Control devices
PR
PUN
INIT
RDR
LINE
SO
Manage system resources
NODE
SE
MAS
RES
LOG
JC
ULOG
Learn about SDSF
TUTOR
HELP
BOOK

Using SDSF

"/d" or DISPLAY is a common use for SDSF:

1. `ulog close` — clear previous user log
2. `ulog` — start new user log
3. `/d tcpip,,netstat,home` — display TCP/IP network information
4. `ulog` — show log

Utilities for JCL

- IEFBR14
- IEBGENR
- IEBCOPY
- IEBDG
- IEBUPDATE

IEFBR14

IEFBR14 is the z/OS equivalent of `/bin/true`:

```
//DSGEN JOB 1,GATES,MSGCLASS=X
//  EXEC PGM=IEFBR14
//A DD DSN=FOO.BAR.DS,DISP=(NEW,CATLG),
//     VOL=SER=WORK02,UNIT=3390,
//     SPACE=(CYL,(3,1,2))
//B DD DSN=BAR.FOO.DS.OLD,DISP=(OLD,DELETE)
```

IEFGENER

Copies one sequential data set to another; the z/OS equivalent of `/bin/cat`:

```
//OGDEN2 JOB 1,BILL,MSGCLASS=X
//  EXEC PGM=IEBGENER
//SYSIN DD DUMMY
//SYSPRINT DD SYSOUT=X
//SYSUT1 DD DISP=SHR,DSN=BILL.SEQ.DATA
//SYSUT2 DD DISP=(NEW,CATLG),DSN=BILL.COPY.DATA,
// UNIT=3390,VOL=SER=WORK02,SPACE=(TRK,3,3))
```

IEBCOPY

- Like IEFGENER for copying, but for PDS(E)s
- Can be restricted to a subset of the PDS's members
- Also used to compress partitioned data sets (PDS)
- Can be used to convert PDS to sequential (backup to tape) and back to PDS (recovery)

IEBDG

- Automatically create data sets with records
- Usually used for generating test data
- Can also be used to mutate existing data sets

IEBUPDTE

Create or modify members in a PDS (like `ar`):

- Changes are explicit (readable) in the arguments to `IEBUPDTE`

$\Rightarrow$ Used for program distribution and patching (like `patch`)

Questions

?