IBM mainframe utility programs

IBM mainframe utility programs are supplied with IBM mainframe operating systems such as MVS to carry out various tasks associated with datasets, etc.

Here follows a list of mainly MVS programs - no VSE or VM utilities are referenced.

History/Common JCL

Many of these utilities were designed by IBM users, through the group SHARE, and then modified or extended by IBM from versions originally written by a user.

These utilities are usually invoked via Job Control Language (JCL). They tend to use common JCL DD identifiers for their data sets:

- SYSIN — input file for the 'commands' for the utility. Often set to DUMMY if the default action is desired.
- SYSUT1 — input file.
- SYSUT2 — output file.
- SYSUT3 — work (spill) file for input (SYSUT1) (often not used).
- SYSUT4 — work (spill) file for output (SYSUT2) (often not used).
- SYSPRINT — output file for printed output from the utility.
- SYSOUT — output file for messages from the utility.
- SYSUDUMP — output file for a system 'dump' if the program fails.

Dataset utilities

IDCAMS

IDCAMS ("Access Method Services") generates and modifies VSAM and Non-VSAM datasets. The "Access Method" reference derives from the initial "VSAM replaces all other access methods" mindset of OS/VS. It probably has the most functionality of all the utility programs, performing many functions, for both VSAM and non-VSAM files. It was intended to replace most of the other dataset utility programs. Example:

```
//XXXXXXXW JOB  XXXXXXX,AAAA,CLASS=G,MSGCLASS=1,NOTIFY=&SYSUID
//STEP001  EXEC PGM=IDCAMS
//SYSIN    DD *
   REPRO INFILE(FILE01) OUTFILE(FILE02)
/*
//FILE01   DD DSN=PROD.FILE1.INPUT,disp=shr   .....
//FILE02   DD DSN=PROD.FILE2.OUTPUT,
//            DISP=(NEW,CATLG,DELETE),
//            UNIT=DASD,
//            SPACE=(TRK,(100,10),RLSE),
//            DCB=(RECFM=FB,BLKSIZE=0,LRECL=80)
//SYSPRINT DD SYSOUT=*
//SYSOUT   DD SYSOUT=*
//SYSUDUMP DD SYSOUT=*
//*
```

In the example above, SYSIN control cards are coming from an in-stream file, but you can instead point to any sequential file or a PDS member containing control cards, if you wish. Example of using SYSIN files would be

something like this:

```
//SYSIN    DD DSN=PROD.MYFILE.REPRO,DISP=SHR
```

or this:

```
//SYSIN    DD DSN=PROD.MYLIB.CNTLLIB(REPRO),
//            DISP=SHR
```

IEBCOMPR

IEBCOMPR compares records in sequential or partitioned data sets.

The IEBCOMPR utility is used to compare two sequential or partitioned datasets. This data set comparison is performed at the logical record level. Therefore, IEBCOMPR is commonly used to verify that a backup copy of a data set is correct (exact match to the original).

During processing, IEBCOMPR compares each record from each data set, one by one. If the records are unequal, IEBCOMPR lists the following information in the SYSOUT:

- The record and block numbers in question.
- The names of the DD statements in which the inconsistency occurred.
- The unequal records.

When comparing sequential data sets, IEBCOMPR considers the data sets equal if the following conditions are met:

- The data sets contain the same number of records.
- The corresponding records and keys are identical.

For partitioned data sets, IEBCOMPR considers the data sets equal if the following conditions are met:

- The directory entries for the two partitioned data sets match - that is, the names are the same, and the number of entries are equal.
- The corresponding members contain the same number of records.
- The corresponding records and keys are identical.

If ten unequal comparisons are encountered during processing, IECOMPR terminates with the appropriate message.

```
//XXXXXXXW JOB   XXXXXXX,AAAA.A.A,CLASS=G,MSGCLASS=1,NOTIFY=XXXXX
//STEP01   EXEC PGM=IEBCOMPR,ACCT=PJ00000000
//     INCLUDE  MEMBER=@BATCHS
//*SYSIN    DD DUMMY
//SYSIN DD *
   COMPARE TYPORG=PO
/*
//SYSUT1   DD DSN=XXXXXXX.OLDFILE,UNIT=DASD,DISP=SHR
//SYSUT2   DD DSN=XXXXXXX.NEWFILE,UNIT=DASD,DISP=SHR
//SYSUT#   DD
```

Note: IEBCOMPR is not a very flexible or user-friendly compare program. It can't restrict the comparison to only certain columns, it can't ignore differences in white space, it doesn't tell you where in the record the difference occurs, and it halts after 10 differences. On the other hand, it is fast, and it is present on all IBM mainframes. So it is very useful in comparing load modules, or checking that a copy worked properly. For comparisons of programs or reports, the ISPF SuperC (ISRSUPC) compare program is often used instead.

IEBCOPY

IEBCOPY copies, compresses and merges partitioned data sets. It can also select or exclude specified members during the copy operation, and rename or replace members.

Some of the tasks that IEBCOPY can perform include the following:

- Creating a backup of a partitioned data set (PDS)
- Copying a PDS in place to reclaim the unused space from deleted members; Also called compressing a PDS.
- Copying selected members to another PDS.
- Renaming selected members of a PDS.
- Merging multiple partitioned data sets into a single PDS.
- Altering, copying and reblocking load modules.

The IEBCOPY utility differs from the other IEB-type utilities in that the DDNAMEs of the input and output DD statements are defined in the user commands as opposed to using the standard SYSUT1 and SYSUT2 DDNAMEs. For the IEBCOPY utility, the required job control statements are as follows:

```
//stepname EXEC PGM=IEBCOPY
//SYSPRINT DD SYSOUT=class
//MYDD1    DD DSN=xxxx.ppp.psps,DISP=shr
//MYDD2    DD DSN=xxxx.ppp.pssp,DISP=shr
//SYSIN    DD *
    COPY INDD=MYDD1,OUTDD=MYDD2
       SELECT MEMBER=(mem1,mem2,mem3)/ EXCLUDE member=(sf,df,sa)
/*
//
```

The MYDD1 and MYDD2 DD statements are names chosen by the user for the partitioned input and output data sets, respectively. You can use any valid DDNAME for these two DD statements. These DDNAMEs are specified in the utility control statements to tell IEBCOPY the name of the input and output data sets.

IEBDG

IEBDG ('Data Generator') creates test datasets consisting of patterned data.

IEBEDIT

IEBEDIT selectively copies portions of JCL.

An example of an IEBEDIT program:

```
//IEBEDITJ JOB ACCT,'',CLASS=P,MSGCLASS=T,MSGLEVEL=(1,1),NOTIFY=&SYSUID
//STEP0001 EXEC PGM=IEBEDIT
//SYSPRINT DD SYSOUT=*
//SYSUT1   DD DSN=xxxxx.yyyyy.zzzzz,DISP=SHR
//SYSUT2   DD SYSOUT=(*,INTRDR)
//SYSIN    DD *
    EDIT TYPE=INCLUDE,STEPNAME=(STEP10,STEP5,STEP15)
/*
//
```

In this example, data set xxxxx.yyyyy.zzzzz should contain sample JCL job(s) (which should include steps named STEP5, STEP10, and STEP15). This IEBEDIT routine copies the selected steps of the sample job onto the SYSUT2

output file (in this example, the internal read execution queue.

The syntax of the EDIT statement is:

```
[label] EDIT [START=jobname]
[,TYPE={POSITION|INCLUDE|EXCLUDE}]
[,STEPNAME=(namelist)]
[,NOPRINT]
```

`START=jobname` specifies the name of the input job to which the EDIT statement applies. Each EDIT statement must apply to a separate job. If START is specified without TYPE and STEPNAME, the JOB statement and all job steps for the specified job are included in the output.

Default: If START is omitted and only one EDIT statement is provided, the first job encountered in the input data set is processed. If START is omitted from an EDIT statement other than the first statement, processing continues with the next JOB statement found in the input data set.

`TYPE={POSITION|INCLUDE|EXCLUDE}` specifies the contents of the output data set. These values can be coded:

`POSITION` specifies that the output is to consist of a JOB statement, the job step specified in the STEPNAME parameter, and all steps that follow that job step. All job steps preceding the specified step are omitted from the operation. POSITION is the default.

`INCLUDE` specifies that the output data set is to contain a JOB statement and all job steps specified in the STEPNAME parameter.

`EXCLUDE` specifies that the output data set is to contain a JOB statement and all job steps belonging to the job except those steps specified in the STEPNAME parameter.

`STEPNAME=(namelist)` specifies the names of the job steps that you want to process.

`namelist` can be a single job step name, a list of step names separated by commas, or a sequential range of steps separated by a hyphen (for example, STEPA-STEPE). Any combination of these may be used in one namelist. If more than one step name is specified, the entire namelist must be enclosed in parentheses.

When coded with TYPE=POSITION, STEPNAME specifies the first job step to be placed in the output data set. Job steps preceding this step are not copied to the output data set.

When coded with TYPE=INCLUDE or TYPE=EXCLUDE, STEPNAME specifies the names of job steps that are to be included in or excluded from the operation. For example, STEPNAME=(STEPA,STEPF-STEPL,STEPZ) indicates that job steps STEPA, STEPF through STEPL, and STEPZ are to be included in or excluded from the operation.

If STEPNAME is omitted, the entire input job whose name is specified on the EDIT statement is copied. If no job name is specified, the first job encountered is processed.

`NOPRINT` specifies that the message data set is not to include a listing of the output data set.

Default: The resultant output is listed in the message data set.

See here for more info.: [1]

IEBGENER

IEBGENER copies records from a sequential dataset, or creates a partitioned dataset..

Some of the tasks that IEBGENER can perform include the following:

- Creating a backup of a sequential data set or a member of a PDS.
- Changing the physical block size or logical record length of a sequential data set.
- Creating an edited data set.
- Printing a sequential data set or a member of a PDS.
- Creating partitioned output data set from sequential input data set.

An example of an IEBGENER program to copy one dataset to another:

```
//IEBGENER JOB  ACCT,'DATA COPY',MSGCLASS=J,CLASS=A
//STEP010  EXEC PGM=IEBGENER
//SYSUT1   DD DSN=xxxxx.yyyyy.zzzzz,DISP=SHR
//SYSUT2   DD DSN=aaaaa.bbbbb.ccccc,DISP=(,CATLG),
//            UNIT=SYSDA,SPACE=(TRK,(5,5),RLSE),
//            DCB=(RECFM=FB,LRECL=1440)
//SYSPRINT DD SYSOUT=*
//SYSIN    DD DUMMY
```

For straight copy tasks, the sort program can often do these faster than IEBGENER. Thus many mainframe shops make use of an option that automatically routes such tasks to the sort ICEGENER program instead of IEBGENER.

On some systems it is possible to send email from a batch job by directing the output to the "`SMTP`" *external writer*. On such systems, the technique is as follows:

```
//IEBGENER JOB  ACCT,'DATA COPY',MSGCLASS=J,CLASS=A
//NORMRC   EXEC PGM=IEBGENER
//SYSPRINT DD SYSOUT=*
//SYSUT1   DD *,LRECL=80
HELO <SYSTEMID>
MAIL FROM:<USERID@SYSTEMID>
RCPT TO:<USERID@SYSTEMID>
DATA
From: <USERID@SYSTEMID>
To: <USERID@SYSTEMID>
Subject: Test Mail

TEST MAIL FROM MAINFRAME
.
QUIT
/*
//SYSUT2   DD SYSOUT=(B,SMTP),LRECL=80
//SYSIN    DD DUMMY
```

It is also possible to attach files while sending the mails from Mainframe.

IEBIMAGE

IEBIMAGE manipulates several types of definitions (AKA *images*) for the IBM 3800 printing subsystem and the IBM 4248 printer. Common uses are for forms control buffers (**FCB**'s), character arrangement tables, character definitions and images of forms to be printed on the output along with the text, for company logos to be printed on the page, or just to print 'graybar' pages (alternating gray & white horizontal backgrounds, to match the previous greenbar paper). With this utility, many different forms or logos could be stored as images, and printed when needed, all using the same standard blank paper, thus eliminating the need to stock many preprinted forms, and the need for operators to stop the printer and change paper.

IEBISAM

IEBISAM unloads, loads, copies and prints ISAM datasets. This is largely obsolete — ISAM has been replaced by VSAM on most modern operating systems; VSAM files use the IDCAMS utility instead of this.

IEBPTPCH

IEBPTPCH ("PrinT and PunCH") prints or punches records from a sequential or partitioned dataset.

Some of the tasks that IEBPTPCH can perform include the following:

- Printing or punching an entire data set, sequential or partitioned (PDS).
- Printing or punching selected PDS members.
- Printing or punching selected records from a sequential or partitioned data set.
- Printing or punching a PDS directory.
- Printing or punching an edited version of a sequential data set or PDS.
- Check for empty dataset

```
//IEBPTPCH JOB
//         EXEC PGM=IEBPTPCH
//SYSIN    DD *
 PRINT     MAXFLDS=2
 TITLE     ITEM=('Name',22),
           ITEM=('GPA',50)
 TITLE     ITEM=(' ',1)
 RECORD    FIELD=(25,1,,22),
           FIELD=(4,51,,50)
/*
//SYSPRINT DD SYSOUT=*
//SYSUT1   DD *
Person 1                307 C Meshel Hall        3.89
Second person           123 Williamson Hall      2.48
3rd person              321 Maag Library         1.52
/*
//SYSUT2   DD SYSOUT=*
//
```

Empty dataset check: If dataset to be checked is empty then RC=4 else 0.

```
//IEBPTPCH JOB
//         EXEC PGM=IEBPTPCH
//SYSUT1   DD DSN=<filename>,DISP=SHR
```

```
//SYSUT2   DD DUMMY,
//         DCB=(BLKSIZE=<block size>,RECFM=FA)
//SYSIN    DD *
 PRINT TYPORG=PS
/*
//SYSPRINT DD SYSOUT=*
//
```

IEBUPDTE

IEBUPDTE ("UPDaTE") incorporates changes to sequential or partitioned datasets. The UNIX `patch` utility is a similar program, but uses different input format markers (*e..g*, "./ INSERT ..." in MVS becomes "@@..." in Unix Patch).

Some programmers pronounce it "I.E.B. up-ditty".

The IEBUPDTE utility is used to maintain source libraries. Some of the functions that IEBUPDTE can perform include the following:

- Creating and updating libraries
- Modifying sequential data sets or PDS members
- Changing the organization of a data set from sequential to partitioned or from partitioned to sequential.

IEBUPDTE is commonly used to distribute source libraries from tape to DASD.

IEBUPDTE uses the same job control statements required by most IEB-type utilities. The only exceptions are as follow:

- IEBUPDTE accepts a PARM parameter coded on the EXEC statement, NEW or MOD. *NEW* indicates that the utility control statements and the input data are contained in the SYSIN DD statement, so no SYSUT1 DD statement is needed. *MOD* indicates that the SYSIN DD statement contains only utility control statements, without input data. Therefore, the SYSUT1 DD statement is required to define the input data set.
- IEBUPDTE reads the input data set from either the SYSUT1 DD statement or from the SYSIN DD statement.

The job control used by IEUPDTE are as follows:

```
//stepname EXEC PGM=IEUPDTE,PARM=NEW
//SYSPRINT DD SYSOUT=class
//SYSUT1   DD ...
//SYSUT2   DD ...
//SYSIN    DD ...
```

Scheduler utilities

IEFBR14

IEFBR14 is a dummy program, normally inserted to JCL when the only desired action is allocation or deletion of datasets.

An example of an IEFBR14 step:

```
//IEFBR14  JOB  ACCT,'DELETE DATASET'
//STEP01   EXEC PGM=IEFBR14
//DELDD    DD DSN=xxxxx.yyyyy.zzzzz,
//            DISP=(MOD,DELETE,DELETE),UNIT=DASD
```

```
```

The calling sequence for OS/360 contained the return address in Register 14. A branch to Register 14 would thus immediately exit the program. However, before and after executing this program, the operating system would allocate & deallocate datasets as specified in the DD statements, so it is commonly used as a quick way to set up or remove datasets.

It consisted initially as a single instruction a "Branch to Register" 14. The mnemonic used in the IBM Assembler was BR and hence the name: IEF BR 14. IEF is, of course, the "prefix" of OS/360's "job management" subsystem.

This single instruction program had an error in it — it didn't set the return code. Hence a second instruction had to be added to clear the return code so that it would exit with the correct status.

There was an additional error reported and fixed by IBM on this now two instruction program. This error was due to the IEFBR14 program not being link-edited as reenterable (simultaneously usable by more than one caller).

Some hackers have taken IEFBR14 and changed the BR 14 instruction to BR 15, thereby creating "the shortest loop in the world", as register 15 contains the address of the IEFBR14 module itself, and a BR 15 instruction would simply re-invoke the module, forever.

System utilities

These utilities are normally used by systems programmers in maintaining the operation of the system, rather than by programmers in doing application work on the system.

ICKDSF

ICKDSF ("Device Support Facility") installs, initializes and maintains DASD, either under an operating system, or standalone.

IEHDASDR

IEHDASDR (“Direct Access Storage Dump and Restore”), an older program not found in the current z/OS manuals, dumps datasets from disk to a printer or backup and restores them from backups.

IEHINITT

IEHINITT ("INITalize Tape") initializes tapes by writing tape labels. Multiple tapes may be labeled in one run of the utility. IBM standard or ASCII labels may be written.

An example of an IEHINITT program:

```
//IEHINITT JOB  ACCT,'LABEL TAPES',MSGCLASS=J,CLASS=A
//STEP0001 EXEC PGM=IEHINITT,REGION=8M
//SYSPRINT DD SYSOUT=A
//LABEL    DD DCB=DEN=2,UNIT=(3490,1,DEFER)
//SYSIN    DD *
LABEL INITT SER=123450,NUMBTAPE=3
/*
```

This example will label 3 tapes on a 3490 magnetic tape unit. Each tape will receive an IBM standard label. The VOLSER will be incremented by one for each tape labeled. Each tape will be rewound and unloaded after being labeled.

IEHLIST

IEHLIST is a utility used to list entries in a Partitioned Dataset (PDS) directory or to list the contents of a Volume Table of Contents (VTOC).

The IEHLIST utility is used to list the entries contained in any one of the following:

- PDS directory
- VTOC
- Catalog (OS CVOL)

An example of an IEHLIST program:

```
//IEHLIST  JOB  ACCT,'LIST PDS',MSGCLASS=J,CLASS=A
//STEP0001 EXEC PGM=IEHLIST,REGION=8M
//SYSPRINT DD SYSOUT=A
//PDS1     DD DSN=xxxx.yyyy.zzzz,DISP=OLD
//SYSIN    DD *
  LISTPDS DSNAME=xxxx.yyyy.zzzz,FORMAT
/*
```

This job will produce a formatted listing of the PDS directory of the PDS named xxxx.yyyy.zzzz.

An example of an IEHLIST program to list a VTOC is very similar:

```
//IEHLIST  JOB  ACCT,'LIST VTOC',MSGCLASS=J,CLASS=A
//STEP0001 EXEC PGM=IEHLIST,REGION=8M
//SYSPRINT DD SYSOUT=A
//VOL1     DD VOL=SER=vvvvvv,DISP=OLD
//SYSIN    DD *
  LISTVTOC VOL=vvvvvv,FORMAT
/*
```

IEHMOVE

IEHMOVE moves or copies collections of data. However, DFSMS (System Managed Storage) environments are now common, and IBM does not recommend using the IEHMOVE utility in those. A move differs from a copy in that during a move the original data set is deleted, or scratched. Some of the tasks that IEHMOVE can perform include the following:

- Moving or copying sequential and partitional data sets
- Moving or copying multi- volume data sets
- Moving an entire volume of data sets

On the surface, IEHMOVE may seen redundant to the IEBGENER and IEBCOPY utilities. However, IEHMOVE is more powerful. The main advantage of using IEHMOVE is that you do not need to specify space or DCB information for the new data sets. This is because IEHMOVE allocates this information based on the existing data sets.

Another advantage of IEHMOVE is that you can copy or move groups of data sets as well as entire volumes of data. Because of the ease in moving groups of data sets or volumes, the IEHMOVE utility is generally favored by system programmers.

A sample IEHMOVE job:

```
//stepname EXEC PGM=IEHMOVE,PARM='LINECNT=xx,POWER=n'
//SYSPRINT DD SYSOUT=class
//SYSUT1   DD UNIT=aaaa,VOL=SER=bbbbbb,DISP=OLD
//anyname1 DD UNIT=cccc,VOL=SER=dddddd,DISP=OLD
//anyname2 DD UNIT=eeee,VOL=SER=ffffff,DISP=OLD
//SYSIN    DD ...
```

The DD statements for IEHMOVE, other than SYSPRINT and SYSIN, refer to DASD or tape volumes instead of individual data sets. However, referencing volumes can pose a problem, since specifying DISP=OLD gains exclusive access to a volume. Therefore, while your IEHMOVE job runs, that entire volume (and all datasets on it) is unavailable to other users. This is acceptable for private volumes, such as tape volumes or mountable volumes, but unacceptable public volumes, such as DASD volumes.

The SYSUT1 DD statement specifies a DASD volume where three work data set required by IEHMOVE are allocated. You must specify unit and volume information for this DD statement.

IEHMOVE was one of the first systems to be developed in PL/S.

IEHPROGM

IEHPROGM builds and maintains system control data. It is also used for renaming and scratching (deleting) a data set.

Some of the tasks that IEHPROGM can perform include the following:

- Deleting (scratching) a data set or PDS member
- Renaming a data set or PDS member
- Cataloging or uncataloging a data set
- Maintaining data set passwords

For cataloging:

```
//SYSIN DD *
   CATLG DSNNAME=data-set-name,
   VOL=device-name=volume-numbeR
/*
//
```

Supporting programs

The following programs are not technically utilities — they are not supplied with the Operating System, but are sold as separate packages. Still, as they are standard items required for programming the computer, nearly all shops will have them installed.

SORT

The Sort/Merge utility is program to sort records in a file into a specified order, or merge pre-sorted files. It is very frequently used; often the most commonly used application program in a mainframe shop. Modern sort/merge programs also can select or omit certain records, summarize records, remove duplicates, reformat records, and produce simple reports. Sort/merge is important enough that there are multiple companies each selling their own sort/merge package for IBM mainframes.

Compilers/Linker

Each programming language used in a computer shop will have an associated compiler that translates a source program into a machine-language object module. Then the object module from the compiler must be processed by the linkage editor, IEWL, to create an executable load module.

IGYCRCTL is a common example of a compiler; it is the compiler for the current IBM Enterprise COBOL for z/OS product. (There have been several previous IBM COBOL compilers over the years, with different names.) There are many other compilers for various other programming languages.

DFSMS

Further information: Hierarchical storage management

DFSMS (System Managed Storage) is a set of programs that allows the operating system itself to take over many of the tasks of managing storage, tasks that were previously performed manually by systems programmers. The *storage administrator* defines *classes* of storage, and *rules* defining dataset assignment into these classes. Then the user (programmer) just needs to specify the *class* for each data file (often by using shop standard naming rules). From then on, the system manages the datasets automatically, taking care of assigning datasets to appropriate storage volumes, providing backup and recovery, migrating datasets up or down between secondary and tertiary storage as needed, and balancing usage of system resources.

References

[1] http://publibz.boulder.ibm.com/cgi-bin/bookmgr_OS390/BOOKS/dgt1u104/5.2.2?SHELF=&DT=19990113105507&CASE=

External links

- IBM: z/OS 1.8 DFSMSdfp utilities manual (http://publibz.boulder.ibm.com/epubs/pdf/dgt2u130.pdf)
- IBM: z/OS 1.8 IDCAMS utility manual (http://publibz.boulder.ibm.com/epubs/pdf/dgt2i250.pdf)
- IBM: z/OS 1.8 ADRDSSU utility manual (http://publibz.boulder.ibm.com/epubs/pdf/dgt2u250.pdf)

Article Sources and Contributors

IBM mainframe utility programs *Source*: http://en.wikipedia.org/w/index.php?oldid=498926537 *Contributors*: Amberroom, Angusus, Apokrif, Arunsingh16, Blaxthos, Bobo192, Canley, Chatul, Chunawalla, Dave6, Essicdo, Flyguy649, Ghettoblaster, Guy Harris, Hgfernan, Hgrosser, Huku-chan, Iefbr14, JYolkowski, Jacobsm, Jauerback, Jbergste, Jeffpc, John Vandenberg, JohnCD, Kiore, Klwtwizzy, Kubanczyk, Mk*, Oxymoron83, Peterh5322, R'n'B, RossPatterson, Rwwww, Samohyl Jan, SchreiberBike, SemperBlotto, Slightsmile, T-bonham, Tabletop, Tcb, TimP, Uryah, Vegaswikian, Waelder, Wikiklrsc, Woohookitty, Zul32, 148 anonymous edits

License