cc-68-44

DECKEDIT Routine for CDC 3300/OS-3, Version 2.0

by G. A. Bachelor

September, 1968

OSU
COMPUTER CENTER
Oregon State University
Corvallis, Oregon 97331

DECKEDIT Routine for CDC 3300/OS-3
Version 2.0

by G. A. Bachelor

Report #cc-68-44

September, 1968
Computer Center
Oregon State University
Corvallis, Oregon 97331

DECKEDIT Routine, Version 2.0 (For CDC 3300/OS-3)

DECKEDIT is a routine which runs under OS-3. It is stored in a public file under the name *DECKEDIT, and may be used by any OS-3 user. Its purpose is to prepare and update (revise) a "library" file, which contains a set of subprograms constituting a single program system. DECKEDIT was written to help manage the set of binary decks constituting OSCAR, and could be useful to other persons or groups who are developing program systems.

A library file (as seen by DECKEDIT) consists of two parts. The first part contains one or more relocatable subprograms, in binary card image form. No two subprograms may have the same name. A file mark terminates this part, which is in the proper form for loading by the standard loader.

The second part of the file contains a directory. Each entry in this directory is a four word BCD record, in which the first two words contain the name of one of the subprograms in the first part. The name consists of one to eight BCD characters, left justified, blank filled on the right. The third and fourth words of the entry contain the date (six BCD characters, left justified) on which the subprogram was placed in the file. The last twelve bits of the fourth word are of no significance in the library file. This field is used as a tag during the processing of a library file. There is one directory entry for each subprogram in the library. The directory entries are usually, but not necessarily, in the same order as the order of the subprograms in part one. The directory is terminated by a file mark. (The only purpose of the directory is to remember the dates associated with the subprograms.)

The DECKEDIT routine can be used to (1) prepare a new library file, (2) add new subprograms to a library, (3) delete subprograms from a library, (4) punch copies of subprograms in a library, or (5) change the order of subprograms in a library.

The DECKEDIT routine goes through the following steps:

1. Reads parameters from the standard input unit (lun 60), to specify the actions desired.

2. Equips library units and punch unit, if necessary. Also removes protection from "new" library unit and from punch unit. Decides whether to write the new library on a scratch unit or directly on the new library unit itself.

3. If there are units containing new subprograms, equips them one at a time and reads from them. Subprograms may be copied to the output, punched, held in memory for later writing, or deleted. If two or more new subprograms have the same name, only the first one is copied to the new library file. The others are deleted.

4. If there is an "old" library file, reads subprograms from it. Subprograms may be copied to the output, punched, held in memory, or deleted. Subprograms read earlier and held in memory may be copied to the output at certain points. Any subprograms whose names are the same as new ones read in step (3) are deleted.

5. If the "old" library has a directory, reads it and obtains dates from it.

6. If the output has been going to a scratch unit, this output is read in and written on the new library unit. (This is done when the old and new libraries are the same unit.)

7. A directory is written on the new library file and then the directory is printed on the standard output (lun 61), in a more readable form.

DECKEDIT also prints a "log" as it goes along. This consists of a list of the names of subprograms read, with an "N" or "O" indicating whether the subprogram is "new" or "old", and one of the notations: (COPIED), if the subprogram was copied to the new library file; (HELD), if it was held in memory; (INSRTED), when a "held" subprogram was written on the new library; or (DELETED), if the subprogram was deleted. The notations (COPIED) and (HELD) are followed by the word PNCH if the subprogram was also written on the punch unit.

Limitations: There is, of course, a limit on memory space for subprograms being held in memory. This limit is about 30,000 words, or 60 file blocks. (Records are stored in memory in variable length form, with one extra word per record to specify the length.) When the old and new library units are the same, enough scratch file space should be allowed for an extra copy of the library.

There is a limit of 128 subprogram names in the table that DECKEDIT uses to keep track of what needs to be done with the various subprograms. If this table becomes full while DECKEDIT is reading subprograms, each subprogram whose name is not already in the table will simply be copied to the new library file, and its name will be omitted from the directory. Also, DECKEDIT will print the message "TABLE OVERFLOW" to let the user know that this condition occurred.

DECKEDIT uses logical units in the range 53 to 59, except unit 56, for handling files. Hence, the user should not use these units.

Parameters.

If DECKEDIT is being used in a batch job, the parameters are key punched (by the user) on one or more cards. The first card must have a 7,8 punch in column 1, then *DECKEDIT, a comma, and the parameters. If more than one card is needed, following cards simply contain additional parameters (no 7,8 punch). The parameters are free field; up to 72 columns may be used, and end of card is equivalent to a space. The parameters are separated by spaces, and terminated by a dollar sign ($).

If DECKEDIT is being used at a teletype, the user types a control statement, starting with *DECKEDIT, then a comma, and the parameters. The parameters may be continued to additional lines, if needed. Each line is ended by [return]. DECKEDIT outputs [line feed]. To end the parameters, the user types a dollar sign and [return].

DECKEDIT recognizes six kinds of parameters, of which the only required parameter is the library (LIB) specification. The forms and meanings of the parameters are described below. The control words (LIB,DELETE,etc.) may be spelled in any way, so long as the first letter is correct. In fact, a single letter is okay (L, D,etc.). The notation (lun or name) indicates that one may specify either a logical unit number in the range 0 to 99, or the name of a saved file. The notation (name1,name2,...) indicates a list of one or more names of subprograms, separated by commas. Parentheses (), commas, slashes (/), and equal signs (=) should be typed as shown in the forms below.

1. Specification of library units.

LIB(lun or name)	Specifies old and new library units (same unit).
LIB(lun1 or name1, lun2 or name2) (lun1 or name1) (lun2 or name2)	Specifies old library and new library
LIB(lun or name/NEW)	Specifies new library unit, no old library.

The LIB parameter must be given, in one of the three forms shown above. Both the new library and the old library (if any) must be files. If the new library is protected, DECKEDIT will remove file protection so that it can write on it. (Exception: If the old and new library units are the same, and there are no new subprograms, no deletions, and no insertions, then it is not necessary to write a new library, and protection is not removed.) If the new library unit is a logical unit number that is not equipped, DECKEDIT will equip the unit as a file. If the new library unit is a name and there is no saved file by that name, DECKEDIT will create a saved file with the specified name. The old library (if any) must exist; DECKEDIT will not create an old library. If the old and new libraries are the same unit, DECKEDIT writes its output on a scratch file until it has finished reading the old library; then this output is copied to the library unit. Library units are rewound before and after using them.

2. Specification of units containing new subprograms.

NEW(lun1 or name1, lun 2 or name2, ...)	Specifies units for new subprograms.
NEW(lun or name/PUNCH, ...)	Same, but punch copies of all subprograms on the new unit.

The NEW parameter is used to specify a list of one or more logical unit numbers or saved files, from which DECKEDIT is to read new subprograms. These subprograms may be used to prepare a new library file, or to revise an existing library. "New" units must be either files, or the card reader. (If a set of decks is to be read from the card reader, it must be terminated by an end of file card). Each new unit is rewound (unless it is a card reader), and subprograms are read from it until a file mark or end of data is encountered.

The PUNCH option may be used on some new units, omitted on others. If there are too many new units to fit their names on one card or line, simply end the parameter with the right parenthesis, and start another NEW parameter on the next line or card.

3. Specification of subprograms to be deleted.

DELETE(name1, name2, ...) Delete the specified subprograms. More than one DELETE parameter may be used, if needed. A DELETE parameter specifies a list of one or more subprogram names. Subprograms with the names given are omitted from the new library. Note: If DECKEDIT encounters more than one subprogram with a particular name, only the first is written on the new library.

Thus, a new subprogram will automatically cause deletion of an old subprogram with the same name. The DELETE parameter deletes all subprograms (including new ones) with the specified names.

4. Specification of punch unit, and subprograms to be punched.

PUNCH= lun or name(name1, name2, ...)	Specifies punch unit (lun or name), and names of subprograms to be punched (name1, name2, ...).
PUNCH(name1, name2, ...)	Specifies subprograms to be punched. Unit 62 will be used unless a unit is specified in some other PUNCH parameter.
PUNCH=lun or name	Specifies punch unit.

PUNCH parameters specify the unit on which to punch subprograms, or the names of subprograms to be punched, or both, as shown above. The punch unit is unit 62 unless otherwise specified. It may be a file or a card punch. If it is a file, DECKEDIT will remove protection from it. If the punch unit is a logical unit number that is not equipped, DECKEDIT will equip it as a file (not a punch). If the punch unit is a name and there is no saved file by that name, DECKEDIT will create a saved file with the specified name.

Subprograms to be punched may be specified by name in the PUNCH parameter, or one can use the PUNCH option in the NEW parameter to cause all subprograms on a new unit to be punched.

Note: DECKEDIT is frequently used just to punch copies of certain subprograms in a library. In this case, only the LIB and PUNCH parameters are used, and DECKEDIT doesn't bother to write a new library. Also, note that "PUNCH" is a bit of a misnomer, since "punched" output can go into a file instead of to a punch.

5. Specification of insertions.

INSERT(name1,name2)	Insert subprogram (name 1) in front of subprogram (name2).
INSERT(name)	Put subprogram (name) at end of library.

INSERT parameters are used to specify rearrangements of the order of subprograms in a library. In the two-name form of the parameters, the first subprogram (either a new one or one in the old library) is held in memory when it is read. Later, when the subprogram with the second name is found (in the old library), the first one is written on the new library. If several subprograms have been held for insertion in front of a particular subprogram, they are all written on the new library, in the order in which their names have occurred in the parameters. Then, the subprogram in the old library is processed; it may be held, deleted, or copied to the new library. If the subprogram with the second name in an INSERT parameter occurs before the first named one, or if there is no subprogram with the second name, then the first subprogram will be held until the end of the old library and then written on the new library. This is also what happens to the subprogram named in the single-name form of INSERT parameter.

Note: By use of INSERT parameters and a little planning, one can rearrange the subprograms in a library into any desired order, subject only to the restriction that all the subprograms to be held will fit in the available memory. (See the earlier paragraph on "limitations".) For example, if the order of subprograms in a library is D, F, A, C, B, E, these can be rearranged into the order A, B, C, D, E, F by the parameters I(C,E) I(D,E) I(F).

6. The "hold" parameter.

HOLD

If the parameter HOLD occurs, then all subprograms read from new units are held in memory, to be inserted in front of old subprograms by the same names (and the old subprograms are, of course, deleted). The effect is to replace old subprograms by new ones with the same names and keep the order of subprograms unchanged. (If HOLD is not used, new subprograms are placed at the beginning of the new library.) New subprograms which are mentioned as the first name in a 2-name INSERT parameter are handled according to the INSERT convention. In other words, HOLD is equivalent to saying INSERT (name,name) for all new subprograms not specifically mentioned. (In fact, this is the way the HOLD parameter is implemented.)

($). Be sure to punch or type a $ to terminate the parameters. The parameters may be given in any order, and should be separated by spaces.

Note 1: Earlier versions of DECKEDIT insisted that a library file have a directory at the end of it. Version 2.0 is not so fussy; any file of binary subprograms can be used as a library file. New libraries written by DECKEDIT will have the directory at the end. A loader library cannot be used as a library for DECKEDIT because it has a loader directory at the beginning (a quite different sort of directory from DECKEDIT's). One can equip a loader library, search forward past file mark (SFPFM), then copy the rest of it to another unit. This unit can then be used as a library for DECKEDIT.

Note 2: EXS cards and RADAR symbol cards may be included in (or at the end of) binary decks. DECKEDIT considers a subprogram deck to start with an IDC card and to continue until the next IDC card, or file mark, or end of data.

Note 3: If a user of DECKEDIT has a (private) file called DIRECTRY, DECKEDIT will equip it and write information on it, giving the names of old and new library units and punch unit, if any. (It does not do this with names of new program units.) This information is in the form required by the DEFINE and DIRECT routines (see separate description). DECKEDIT doesn't try very hard to do this; if there is no file called DIRECTRY, or if there is one but it is busy or protected, DECKEDIT simply drops the idea and goes on with its main job. The purpose of this feature is to aid those users who are using DEFINE and DIRECT to help maintain a list of names of saved files.

Note 4: Earlier versions of DECKEDIT had a limit of 4 units for new subprograms. This restriction has been removed in Version 2.0. Each new unit specification requires three words of memory, and there is a lot of memory space available.

<u>Use of DECKEDIT in batch jobs</u>.

(1) Given a set of binary decks of subprograms, to prepare a new library file of them, to be called PROGLIB.

$\overset{7}{8}$JOB,77777,ABC,SAVE FOR ABC
$\overset{7}{8}$*DECKEDIT,L(PROGLIB/N) N(60) $
  [Binary decks of subprograms to be placed in library.]
$\overset{77}{88}$ [end of file card]
$\overset{7}{8}$LOGOFF

(2) Revise a library (PROGLIB), with new subprograms on binary decks. Also load and run the new program library.

$\overset{7}{8}$JOB,77777,ABC,SAVE FOR ABC
$\overset{7}{8}$*DECKEDIT,L(PROGLIB) N(60) $
  [Binary decks of new subprograms.]
$\overset{77}{88}$ [end of file]
$\overset{7}{8}$EQUIP,40=PROGLIB
$\overset{7}{8}$LOAD,40
MAP
RUN
  [data for program]
$\overset{7}{8}$LOGOFF

(3) Assemble and punch some new subprograms, use them together with some binary decks to update a library (PROGLIB), putting the new library on a scratch unit, and load and run it.

$\overset{7}{8}$JOB,77777,ABC,SAVE FOR ABC
$\overset{7}{8}$LABEL,62/ABC
$\overset{7}{8}$COMPASS,L,R,X,P
  [source decks for assembly]
  FINIS
$\overset{7}{8}$*DECKEDIT,L(PROGLIB,44) N(56,60) $
  [binary decks]
$\overset{77}{88}$
$\overset{7}{8}$LOAD,44

MAP
RUN
   [data]
$^{7}_{8}$LOGOFF

(4) Update a library (PROGLIB), creating and saving a new library file, to be called NEWPLIB. New subprograms are on saved files named NEW1 and NEW3. All subprograms on NEW1 are to be punched. Also punch a copy of subprogram named PROG3, delete subprograms named PROG6 and PROG8 (all on PROGLIB), and insert subprogram named PROG5 in front of subprogram named PROGX, in preparing new library. (Note: PROGLIB remains unchanged.)

$^{7}_{8}$JOB,77777,ABC,SAVE FOR ABC
$^{7}_{8}$LABEL,62/ABC
$^{7}_{8}$*DECKEDIT,L(PROGLIB,NEWPLIB) N(NEW1/P,NEW3)
I(PROG5,PROGX) P(PROG3) D(PROG6,PROG8) $
$^{7}_{8}$LOGOFF

Use of DECKEDIT from a teletype.

In using DECKEDIT from a teletype, one would do the same kinds of things as in the batch jobs above, except that binary decks would not be read from unit 60 (the teletype). One would type control statements such as:

(1) #*DECKEDIT,L(PROGLIB,44) N(56) $ [return]
(2) #*DECKEDIT,L(PROGLIB,NEWPLIB) N(NEW1/P,NEW3) [return]
I(PROG5,PROGX) P(PROG3) D(PROG6,PROG8) $ [return]

The user ends each line with the return key. DECKEDIT (or OS-3) supplies the line feed. After reading a line with a dollar sign, DECKEDIT goes to work, It first prints a heading, then begins printing the log and directory, as described earlier. If the teletype user does not wish to have all of this printed out, he can use the "break" key to stop the output, then type the control statement #MI [return]. DECKEDIT will finish its job without printing anything except a couple of blank lines, or possibly an error message. When done, DECKEDIT puts the user back into OS-3 control mode (# is printed out).

If one makes a mistake while typing the parameters for DECKEDIT, one can use the reverse slant (\) to cancel preceding characters. However, one cannot cancel errors in a line which has already been ended by "return". Also, one must spell *DECKEDIT correctly. If an error in either of these situations is made, one must type [control A], then start over.

DECKEDIT error messages.

There is one situation in which DECKEDIT prints an "informative" message. This is TABLE OVERFLOW, which is printed if DECKEDIT encounters more than 128 subprogram names. The occurrence of this message means that the directory will not contain the names of all the subprograms.

All other errors detected by DECKEDIT cause an abort; DECKEDIT prints the message DECKEDIT ABORT , followed by one of the messages listed below. These messages should be self-explanatory, for the most part. "PARAMETER ERROR" occurs if one of the parameters does not start with D, H, I, L, N, or P; if a parameter has an improper form; or if there is no LIB parameter. "CANNOT RFP" means that DECKEDIT cannot remove file protection from the unit mentioned. "INPUT UNIT" refers to one of the units containing new subprograms. "SCRTCH UNIT" refers to the scratch file on which DECKEDIT writes the library temporarily, when the old and new libraries are the same unit. There should not be any BCD cards on this unit since DECKEDIT writes only binary records on it. "*SYSTEM ERROR*" should not occur; it indicates a malfunction either in DECKEDIT or in OS-3. The error messages are listed in approximately the order in which the conditions are likely to be encountered. If more than one of these error conditions is present, the first one discovered causes an abort, with the corresponding error message.

```
PARAMETER ERROR
NEW LIB IS BUSY
NEW LIB NOT A FILE
CANNOT RFP NEW LIB
OLD LIB IS BUSY
OLD LIB DOES NOT EXIST
OLD LIB NOT A FILE
PUNCH UNIT IS BUSY
CANNOT RFP PUNCH UNIT
INPUT UNIT IS BUSY
INPUT UNIT DOES NOT EXIST
INPUT UNIT NOT A FILE OR CARD RDR
BCD CARD ON INPUT UNIT
BCD CARD ON OLD LIB
MEMORY FULL
BCD CARD ON SCRTCH UNIT
*SYSTEM ERROR*
```

Appendix

List of DECKEDIT parameters for handy reference.

LIB(lun or name)	Old and new libs (same).
LIB(lun1 or name1, lun2 or name2)	Old and new libs.
LIB(lun or name/NEW)	New lib (no old lib).
NEW(lun1 or name1, lun2, or name2,...)	Units for new subprograms.
NEW(lun or name/PUNCH,...)	Same, but punch copies.
DELETE(name1, name2,...)	Delete specified subprograms.
PUNCH=lun or name (name1, name2, ...)	Specifies punch unit and subprograms to be punched.
PUNCH(name1,name2, ...)	Punch specified subprograms.
PUNCH=lun or name	Specifies punch unit.
INSERT(name1, name2)	Insert subprogram (name1) in front of subprogram (name2).
INSERT(name)	Put subprogram (name) at end of lib.
HOLD	Hold all new subprograms in memory, insert them in lib at places where old subprograms by same names are.
$	End of parameters.