### EMPLIB: A SEQUENTIAL FILE PROGRAM LIBRARIAN

EMPLIB, written for use on a CDC 6000 computer operating under Scope 3, is a librarian program whose function is to maintain an active library and a separate permanent archive of program UPDATE and object files on a sequential storage device such as a magnetic tape reel. The EMPLIB librarian can perform readout or alteration of the library or archive, and also certain file-positioning actions and program object file editing.
<table>
<thead>
<tr>
<th>KEY WORDS</th>
<th>LINK A</th>
<th>LINK B</th>
<th>LINK C</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>ROLE</td>
<td>WT</td>
<td>ROLE</td>
</tr>
<tr>
<td>Computer Programs</td>
<td>8</td>
<td>3</td>
<td></td>
</tr>
<tr>
<td>Librarians</td>
<td>8</td>
<td>3</td>
<td></td>
</tr>
<tr>
<td>Information Retrieval</td>
<td>8</td>
<td>3</td>
<td></td>
</tr>
<tr>
<td>Utility Programs</td>
<td>8</td>
<td>3</td>
<td></td>
</tr>
<tr>
<td>FORTRAN</td>
<td>8</td>
<td>3</td>
<td></td>
</tr>
</tbody>
</table>
HDL-TR-1591

EMPLIB: A SEQUENTIAL FILE PROGRAM LIBRARY

by

William T. Wyatt, Jr.

April 1972

This work was sponsored by the Defense Nuclear Agency under subtasks EA091 and EA094.

U.S. ARMY MATERIEL COMMAND,

HARRY DIAMOND LABORATORIES

WASHINGTON, D.C. 20438

APPROVED FOR PUBLIC RELEASE; DISTRIBUTION UNLIMITED.
ABSTRACT

EMPLIB, written for use on a CDC 6000 computer operating under Scope 3, is a librarian program whose function is to maintain an active library and a separate permanent archive of program UPDATE and object files on a sequential storage device such as a magnetic tape reel. The EMPLIB librarian can perform readout or alteration of the library or archive, and also certain file-positioning actions and program object file editing.
1. INTRODUCTION

EMPLIB is a program written in CDC Fortran Extended and Compass for use on CDC 6000-series computers operating under Scope 3. It has been tested and run under Scopes 3.2 and 3.3, and requires about 54,200 words (octal) to load and execute. EMPLIB is a librarian program whose function is to maintain a library of frequently used program UPDATE files (called "source" files here) and program object files (called "binary" files here, i.e., the compiler or assembler object output). The term "file" is defined here as a string of data terminated by an EOF. The library is kept on a magnetic tape or other permanent sequential data storage device. EMPLIB also maintains an archive magnetic tape of program source or binary files to be saved indefinitely. The user may run the librarian program EMPLIB and cause it to perform certain library or archive functions by placing directive cards in the input card stream to be read by the librarian. These directive cards are processed sequentially, allowing library alteration, program file readout, user-assigned filenames for readin and readout functions, certain filename actions such as rewind, endfile, and skipfile, and archive additions or readout. The term "filename" is defined here as a logical file name (i.e., LGO, TAPE1, OLDPL, etc.).

2. THE LIBRARIAN

The librarian uses nine working filenames for various functions. All functions but one are assigned a one- or two-letter mnemonic and are associated by default with certain filenames which may be altered by the user during execution of the librarian. The file functions, mnemonics, default filenames, and purpose are listed below:

<table>
<thead>
<tr>
<th>Function</th>
<th>Mnemonic</th>
<th>Filename</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td>card input</td>
<td>I</td>
<td>INPUT</td>
<td>Contains EMPLIB directives.</td>
</tr>
<tr>
<td>print output</td>
<td>O</td>
<td>OUTPUT</td>
<td>Contains printed output.</td>
</tr>
<tr>
<td>library</td>
<td>L</td>
<td>EMPLIB</td>
<td>Contains the program library.</td>
</tr>
<tr>
<td>archive</td>
<td>A</td>
<td>ARCHIV</td>
<td>Contains the program archive.</td>
</tr>
<tr>
<td>source input</td>
<td>SI</td>
<td>NEWPL</td>
<td>Source files read from NEWPL.</td>
</tr>
<tr>
<td>source output</td>
<td>SO</td>
<td>OLDPL</td>
<td>Source files written to OLDPL.</td>
</tr>
<tr>
<td>binary input</td>
<td>BI</td>
<td>LGO</td>
<td>Binary files read from LGO.</td>
</tr>
<tr>
<td>binary output</td>
<td>BO</td>
<td>XQT</td>
<td>Binary files written to XQT.</td>
</tr>
<tr>
<td>scratch</td>
<td>none</td>
<td>TAPE40</td>
<td>Scratch file for librarian.</td>
</tr>
</tbody>
</table>

All of the file functions, with the exception of the scratch function, may be assigned different filenames by use of the FILES directive described later. The filenames accessed by the librarian must all be odd-parity files as distinct from even-parity BCD files. The librarian can, of course, access an odd parity file onto which a BCD file has been copied. The terms file and binary file as used here both refer to files with odd parity. The difference between the two types of files is one of name only, and is conventionalized so that program UPDATE files are designated as source files and program object files are designated as binary files. The directives SELECT and REFUSE work properly only with binary files that are, in fact, program object files. Otherwise, any data file may be treated as a program source or binary file and manipulated by the librarian. The first file on the library filename is intended to be the librarian program object file, where it may be easily copied off and executed. (If the library tape is executed directly, the system loader will unload the tape, preventing later access to the library.) The second file contains a table of the library contents. Subsequent files are source and binary files previously placed in the library. Each file is identified in the table of contents by a name identifier, a version identifier, a mode identifier (to distinguish whether it is a source file or a binary file), and date of entry into the library. The name and version...
Identifiers must be from one to ten characters with no imbedded blanks or commas. The version identifier is optional and will be all blanks if not specified by the user. The archive filename contains two data files for each source or binary file kept on it. The first is a file containing the table of contents information about the source or binary file, and the second data file is the source or binary file itself. The end of the archive is denoted by a file containing just the one word "LAST."

3. LIBRARIAN DIRECTIVES

The various functions the librarian can perform will be illustrated through their use in the following examples. The completed output is listed in Appendix A. It is assumed in the first example that the librarian object file has been copied to the filename EMPLIB (so as to allow creation of the library) and that a magnetic tape has been assigned to the filename ARCHIV. Execution of the librarian causes the filename EMPLIB to be rewound if the first directive is not a FILES directive; thus the library filename must be changed immediately if it is not to be EMPLIB.

Directives are free-field, but must have a dollar sign in column one. Directives and identifiers must be separated by blanks, unless commas are required. The librarian will copy each directive card to the print file as encountered and then add a description of any action taken. On the print file, directives can be recognized by the single dollar sign, whereas statements originated by the librarian begin with "EMPLIB $$."

3.1 CREATE

CREATE causes a library to be created on the filename attached to the library function. Physically, the first file is skipped on the library filename and a table of contents file is written which records the first file as "EMPLIB" and the second file as "TOC." The library must be created (establishing a table of contents) before any library additions can be performed. In fact, a table of contents is required by all but the following directives: CREATE, CREATEARCH, FILES, SKIP, SKIPB, HISTORY, ENDFILE, REWIND, FIND, AND FINDB. The directives SELECT and REFUSE may or may not require a table of contents. $CREATE

3.2 CREATEARCH

CREATEARCH causes an archive to be established on the filename attached to the archive function. Physically, the hollerith word "LAST" is written on the archive. The archive must be established before any archive additions can be performed. The archive is rewound before and after the creation. $CREATEARCH

3.3 End of librarian input

The sequence of directives is terminated by a 7-8-9 card. If the last operation on the source output or binary output filename was a write-end-of-record, the filename is EOF'd and backspaced before execution is ended. All the following directives may be given in the same or any subsequent execution of the librarian once the library has been created. The file name/versions specified must be in the library when the directive is processed by the librarian, except for new name/versions in ADD (ADDB) and RENAME (RENAMEB), and
except for the archive directive FIND (FINDB). File name/versions appearing with the FIND (FINDB) directive must already be on the archive when the directive is processed.

3.4 ADD and ADDB

ADD (ADDDB) causes the source (binary) file on the source (binary) input filename to be added to the library, and assigns it to the name and version specified. The source (binary) input filename is rewound before reading is begun, unless suppressed by a NOREWIND directive (discussed later). $ADD PROG VERS

3.5 TOC

TOC causes a table of contents of the library to be printed. $TOC

3.6 FILES

FILES causes the file functions whose mnemonics are specified on the directive card to be reassigned different filenames. A reassignment consists of the mnemonic, one or more blanks, and the new filename, in that order. Multiple reassignments must be separated by commas. Old filenames whose last operation was a write-end-of-record are EOF'd and back-spaced before being detached from a file function when the reassignment is made. This directive may be issued even if the library has not been created. $FILES SI OLD, BI AGO, SO NEW

3.7 SKIP and SKIPB

SKIP (SKIPB) causes the number of files specified to be skipped in a forward direction on the source (binary) input filename. Up to 999 files may be skipped with one directive. If the number of files to be skipped is not specified, one file is skipped. $SKIPB 2

3.8 NOREWIND

NOREWIND suppresses the automatic rewind of the source (binary) input filename for the next (and only the next) ADD (ADDB) or CHANGE (CHANGEB) directive encountered. $NOREWIND

3.9 CHANGE and CHANGEB

CHANGE (CHANGEB) causes the source (binary) file name/version specified to be replaced on the library by the next file encountered on the source (binary) input filename. The filename is automatically rewound before reading unless suppressed, as in this example, by a NOREWIND directive. The present data is placed in the library table of contents for the file changed. The file changed must already be in the library. $CHANGE PROG VERS
3.10 **RENAME and RENAMEB**

RENAME (RENAMEB) causes the first source (binary) file name/version given on the card to be renamed in the table of contents file with the second source (binary) file name/version given on the card. The first and second file name/version must be separated by a comma. $RENAME PROG VERS, PROGA NEWNAME

3.11 **DROP and DROPB**

DROP (DROPB) causes the source (binary) file name/version to be removed from the library and its entry in the table of contents file to be deleted. The first file on the library (the EMPLIB binary file) will never be dropped, since this will cause the library to be scrambled. $DROP NEWPROG

3.12 **KEEP and KEEPB**

KEEP (KEEPB) causes the source (binary) file name/version specified to be added to the archive. The specified file name/version must already be in the library. Once added to the archive, a file cannot be removed from the archive by the librarian. $KEEPB PROG VERS

3.13 **HISTORY**

HISTORY causes the contents of the archive to be scanned and a list of the file name/versions encountered to be printed. This directive may be processed by the librarian even if the library has not been created; only the archive need exist. $HISTORY

3.14 **RUN**

RUN causes the first binary file on the library with the specified name to be copied to the binary output filename irrespective of the program version. Thus, only the program name need be specified. The terminating EOF is not copied, so further material may be copied to the binary filename to complete the desired load module. $RUN PROG

3.15 **COPY and COPYB**

COPY (COPYB) causes the source (binary) file name/version specified on the library to be copied to the source (binary) output filename. The terminating EOF is not copied, just as for the RUN directive. $COPY PROGA NEWNAME

3.16 **SELECT**

SELECT causes the specified binary file name/version on the library to be scanned for the named object programs or subprograms, which are copied as encountered to the binary output filename with no terminating EOF. On the directive card the binary file name/version must be the first identifiers after the SELECT word, followed by a comma, and then followed by the program or subprogram names separated by commas. If the file name/version is
omitted so that the next non-blank character after the directive is a comma, the next file on
the binary input filename will be scanned for the named programs and subprograms; this ac-
tion does not require a table of contents. If the last non-blank character on the card is a
comma, continuation cards will be read until the final non-blank card character is not a comma.
Continuation cards must not contain a dollar sign in column one, and must contain information
in columns one through 79 only. Up to 100 program or subprogram names may be specified in
the directive. A list of all object routines encountered and their selection or refusal is
printed. The largest object routine that can be processed by SELECT or REFUSE must be
less than 6000 words long. A statement of the maximum size processed is printed at end of
execution. $SELECT PROG VERI, ISO, SPLITR, SPLITC.

$SELECT, ISO, SPLITR, SPLITC.

3.17 REFUSE

REFUSE causes the same action as SELECT, except that specified object program and
subprogram names are not copied to the binary output filename and all others encountered
are copied. Empty records are not copied. Up to 100 names may be specified for re-


3.16 ENDFILE

ENDFILE causes the file function whose mnemonic is specified to have an EOF written
on the filename assigned to the file function. Only one file function may be specified on the
directive card and only the file functions BO and SO may be endfiled with this directive.
$ENDFILE FO

3.19REWIND

REWIND causes the file function whose mnemonic is specified to have its assigned file-
name rewound. If information had been written to the filename, end-of-information termina-
tors are written to the filename before it is rewound. The file functions I, O, and L cannot be
rewound with this directive. $REWIND SO

3.20 FIND and FINDB

FIND (FINDB) causes the archive to be searched for the source (binary) file name/
version specified, which is then copied to the source (binary) output filename. No EOF is
written, just as for the COPY (COPYB) directive. The directives FIND and FINDB may be
processed by the librarian even if the library has not been created; only the archive is re-
quired to exist. $FINDB PROG VERS

3.21 REPLACE and REPLACEB

REPLACE (REPLACEB) causes the source (binary) file name/version specified to be
replaced on the library on the next file encountered on the source (binary) input filename,
and given a new name/version label. This combines the functions of CHANGE (CHANGEB)
and REPLACE (REPLACEB). The directive format is the same as for the RENAME
(RENAMEB) directive. The source (binary) input filename is rewound before reading is
begun, unless suppressed by a NOREWIND directive.

4. LIBRARIAN ERROR MESSAGES

When the librarian detects an error involving the directive card being processed, a mes-
sage describing the nature of the error is printed and the rest of the librarian card input file
is copied to the print output file, after which execution is terminated by a call to the nonexistent
subroutine ABORT which causes a mode one (address-out-of-range) error termination.
Terminators are assured to be on any source or binary output filename if the filename has been written on, just as for a normal termination.

If another kind of error is detected, an informative message is printed and execution is terminated immediately with a CALL ABORT. Terminators are not assured for filenames assigned to output functions at the time the error was detected.

4.1 Directive format errors

The following errors use the ABORT termination after checking file terminators:

1. Missing or misplaced dollar sign on directive card. The dollar sign must be in column one.

2. Improper directive. A directive cannot be found on the card.

3. Unrecognizable directive. The specified directive is not familiar to the librarian.


5. Missing program filename. A program file name cannot be found on the card when one is required.

6. Program file name too long. The specified program file name is longer than 10 characters.

7. Program file version too long. The specified program file version is longer than 10 characters.

8. Program file name/version not in table of contents. The specified name/version is not on the library and the directive cannot be executed.

9. Adding file already in table of contents. The specified name/version/mode is already in the library; a unique name/version/mode must be specified.

10. Missing comma. A needed comma is missing between the old name/version and the new name/version on a RENAME, RENAMES, REPLACE, or REPLACEB directive.

11. Word is too long. A word is longer than 10 characters on a FILES directive card. In fact, SCOPE can handle filenames only up to seven characters long, so care should be taken not to use 8, 9, or 10 character filenames.

12. Unrecognized file type. The file function type specified is not recognized.

13. More than 100 record names. Too many program and subprogram names are listed in a SELECT or REFUSE directive.


15. Illegal directive for the file type. The directive is not allowed for the file function type specified.
16. Illegal number. Unrecognizable number on a SKIP or SKIPB card; 999 is the maximum allowed.

17. Program file name/version not on archive. The name/version specified by a FIND or FINDB directive is not in the archive.

4.2 Other errors

The following errors cause an immediate CALL ABORT termination:

1. KEEP read parity error. A read parity error occurred while reading the library for a KEEP or KEEPB directive.

2. KEEP write parity error. A write parity error occurred while writing to the archive for a KEEP or KEEPB directive. The former contents of the archive are intact, but an end-of-archive record no longer exists.

3. FIND read error. A read parity error occurred while reading the archive for a FIND or FINDB directive.

4. HISTORY read error. A read parity error occurred while reading the archive for a HISTORY directive.

5. GETTOC parity error. A read parity error occurred while the librarian was trying to read the table of contents file.

6. Empty file. The filename specified as the location of a program file was empty.

7. CPYFIL read parity error. A read parity error occurred while the librarian was skipping a file.

8. I/O error in CPYBUF. An I/O error occurred while the librarian was copying a file.

9. End-of-information encountered. An EOI was encountered while trying to copy a file; i.e., the filename was short-terminated.

10. TOC write parity error in PFWFIL. A write parity error occurred while the librarian was writing the table-of-contents file to the library for a library alteration directive.

11. Read error in CPYREC. A read parity error occurred while the librarian was reading or binary input filename during processing of a SELECT or REFUSE directive.

12. Write error in CPYREC. A write parity error occurred while the librarian was copying a program or subprogram record to the binary output filename during processing of a SELECT or REFUSE directive.

5. USER HINTS

The following information will be useful to the EMPLIB user.
5.1 List of directives

PROG and PROGA are program file names, and VERS and VERSA are program file versions in the following examples. Items enclosed in parenthesis are optional. An asterisk denotes the file is rewound before reading unless suppressed by a NOREWIND directive.

<table>
<thead>
<tr>
<th>Input</th>
<th>Output</th>
<th>Directive</th>
</tr>
</thead>
<tbody>
<tr>
<td>L</td>
<td>L</td>
<td>$CREATE</td>
</tr>
<tr>
<td></td>
<td>A</td>
<td>$CREATEARCH</td>
</tr>
<tr>
<td>SI*</td>
<td>L</td>
<td>$ADD PROG (VER)</td>
</tr>
<tr>
<td>BI*</td>
<td>L</td>
<td>$ADDB PROG (VERS)</td>
</tr>
<tr>
<td>L</td>
<td></td>
<td>$TOC</td>
</tr>
<tr>
<td></td>
<td></td>
<td>$FILES BI ABC, A PQR7826</td>
</tr>
<tr>
<td>SI</td>
<td></td>
<td>$SKIP (5)</td>
</tr>
<tr>
<td>BI</td>
<td></td>
<td>$SKIPB (999)</td>
</tr>
<tr>
<td>SI*</td>
<td>L</td>
<td>$CHANGE PROG (VERS)</td>
</tr>
<tr>
<td>BI*</td>
<td>L</td>
<td>$CHANGEB PROG (VERS)</td>
</tr>
<tr>
<td>L</td>
<td>L</td>
<td>$RENAME PROG (VERS), PROGA (VERSA)</td>
</tr>
<tr>
<td>L</td>
<td>L</td>
<td>$RENAMEB PROG (VERS), PROGA (VERSA)</td>
</tr>
<tr>
<td>L</td>
<td>L</td>
<td>$DROP PROG (VERS)</td>
</tr>
<tr>
<td>L</td>
<td>A</td>
<td>$DROPB PROG (VERS)</td>
</tr>
<tr>
<td>L</td>
<td>A</td>
<td>$KEEP PROG (VERS)</td>
</tr>
<tr>
<td>A</td>
<td></td>
<td>$KEEPB PROG (VERS)</td>
</tr>
<tr>
<td>A</td>
<td></td>
<td>$HISTORY</td>
</tr>
<tr>
<td>L</td>
<td>BO</td>
<td>$RUN PROG</td>
</tr>
<tr>
<td>L</td>
<td>SO</td>
<td>$COPY PROG (VERS)</td>
</tr>
<tr>
<td>L</td>
<td>BO</td>
<td>$COPYB PROG (VERS)</td>
</tr>
<tr>
<td>L</td>
<td>BO</td>
<td>$SELECT PROG (VERS), SUBA, SUBB, SUBC</td>
</tr>
<tr>
<td>BI</td>
<td>BO</td>
<td>$SELECT, SUBA, SUBB, SUBC</td>
</tr>
<tr>
<td>L</td>
<td>BO</td>
<td>$REFUSE PROG (VERS), SUBA, SUBB, SUBC</td>
</tr>
<tr>
<td>BI</td>
<td>BO</td>
<td>$REFUSE, SUBA, SUBB, SUBC</td>
</tr>
<tr>
<td></td>
<td>BO or SO</td>
<td>$ENDFILE BO</td>
</tr>
<tr>
<td>All but I,O,L</td>
<td></td>
<td>$REWRITE SI</td>
</tr>
<tr>
<td>A</td>
<td>SO</td>
<td>$FIND PROG (VERS)</td>
</tr>
<tr>
<td>A</td>
<td>BO</td>
<td>$FINDB PROG (VERS)</td>
</tr>
<tr>
<td>SI*</td>
<td>L</td>
<td>$REPLACE PROG (VERS), PROGA (VERSA)</td>
</tr>
<tr>
<td>SI*</td>
<td>L</td>
<td>$REPLACEB PROG (VERS), PROGA (VERSA)</td>
</tr>
</tbody>
</table>

All directives except CREATE which use the library (L) as input or output require a created library. All directives except CREATEARCH which use the archive (A) as input or output require a created archive.

5.2 File Actions

The librarian checks the first directive encountered and, if it is not FILES directive, rewinds the library (which has the filename EMPLIB) and looks to see if a table of contents exists. If it is a FILES directive, rewinding the library file is deferred to just prior to processing the next directive.

All directives which use the library as output cause the entire library to be copied to the scratch filename TAPE40 and recopied in its modified form back to the library filename. If the library is of substantial length and if more than one or two directives of this kind are to
be processed, much PP time will be saved if the library tape is copied to a disk filename be-
fore librarian execution and then recopied from the disk filename back to the library tape after
librarian execution. The library filename must be the disk filename, of course. This also
helps protect the library tape from write parity errors.

A good practice is periodically to copy the entire library and the entire archive to a back-
up library tape and a backup archive tape, to avoid loss of program files if the first-line
copies are impaired by permanent parity errors.

If the library is of short length, it may be practical to have the library reside on a
permanent disk file instead of on a magnetic tape. The archive will generally be too large
for this, however.

5.3 Examples of usage

Although it would not be possible to illustrate all the possible uses of the EMPLIB li-
brarian, a few examples will be useful to convey the flexibility and simplicity of the program.
The examples are for a Scope 3.3 system. All TOC directives are optional, but are recom-

1. Update, compilation of changes, and execution.

JOB, CM54000, TP1
REQUEST, EMPLIB, (540/NORING)
COPYBF (EMPLIB, LIB, 1)
LIB.
RETURN (EMPLIB)
UPDATE (P)
FTN (I=COMPLE)
REWIND (XQT)
COPYBF (XQT, LGO, 1)
LGO.
7-8-9
$TOC
$COPY NEPHI CORRQ
$REFUSE NEPHI CORRQ, PHOTON, GROUND
7-8-9
(Update input with changes for subroutines PHOTON and GROUND.)
7-8-9
(Input data.)
6-7-8-9

This could also be accomplished by the following cards between the FTN card and UP-
DATE input cards:

LGO.
7-8-9
$TOC
$FILES BO LGO
$COPY NEPHI CORRQ
$REFUSE NEPHI CORRQ, PHOTON, GROUND
7-8-9
2. Update, compilation of changes, and alteration of library.

\begin{verbatim}
JOB, CMS4000, TP1.
REQUEST, EMPLIB (540/RING)
COPYBF (EMPLIB, LIB, 1)
LIB.
UPDATE (P, N, W) (W makes new UPDATE library sequential.)
FTN (I = COMPILE)
LIB.
UNLOAD (EMPLIB)
$TOC
COPY NEPHI CORRQ
FILES BO LGO
REFUSE NEPHI CORRQ, PHOTON, GROUND
7-8-9
(Update input with changes for subroutines PHOTON and GROUND.)
7-8-9
$DROP NEPHI CORRQ
$DROPB NEPHI CORRQ
$ADD NEPHI CORRR
$ADDB NEPHI CORRR
$TOC
6-7-8-9
\end{verbatim}

More efficient use of the greater speed of disk files would be made by using the following control cards in the previous example:

\begin{verbatim}
JOB, CMS4000, TP1.
REQUEST, ZAP. (540/RING)
COPYBF (ZAP, EMPLIB, 100) (less than 100 files on ZAP)
EMPLIB.
UPDATE (P,N,W)
FTN (I = COMPILE)
EMPLIB.
REWRIND (EMPLIB, ZAP)
COPYBF (EMPLIB, ZAP, LGO)
UNLOAD (ZAP)
7-8-9
\end{verbatim}

3. Execution of one program.

\begin{verbatim}
JOB, CMS4000, TP1.
REQUEST, EMPLIB (540/NORING.)
COPYBF (EMPLIB, LIB, 1)
LIB.
RETURN (EMPLIB)
RFL,100000.
REDUCE.
XQT.
7-8-9
$RUN NEPHI
7-8-9
\end{verbatim}
4. Execution of several programs.

JOB, CM54000, TP1.
REQUEST, EMPLIB (540/NORING.)
COPYBF (EMPLIB, LIB, 1)
LIB.
RETURN (EMPLIB)
XQT.
NEXT.
LAST.
7-8-9
$TOC
$COPYB PROG FIRST
$FILES BO NEXT
$RUN PROGSEC
$FILES BO LAST
$RUN PROGFIN
7-8-9
(Data for PROG/FIRST.)
7-8-9
(Data for PROGSEC.)
7-8-9
(Data for PROGFIN.)
6-7-8-9

(Input data for NEPHI.)
6-7-8-9
Appendix A. SAMPLE OUTPUT
EMPLIB $8 THF DATE IS 10/29/71 AND THE WORK FILES ARE
BINARY OUTPUT = NOT
SOURCE OUTPUT = OLDPL
EMPLIB LIBRARY = EMPLIB
EMPLIB OUTPUT = EMPLIB
$ADD PROG VERS
EMPLIB $8 ADDED 9TH FILE (SOURCE PROG VERS 10/29/71 ) TO EMPLIB FROM EMPLIB FILE.
$ADD PROG VERS
EMPLIB $8 ADDED 9TH FILE (BINARY PROG VERS 10/29/71 ) TO EMPLIB FROM LGO FILE.
$TOC
EMPLIB $8 TABLE OF CONTENTS OF EMPLIB
1 EMPLIB
2 TOC
3 PROG VERS
4 PROG VERS
$FILES $1 OLD, $2 AGO, $3 NEW
EMPLIB $8 MADE SOURCE INPUT FILE OLD INSTEAD OF EMPLIB *
EMPLIB $8 MADE BINARY INPUT FILE AGO INSTEAD OF LGO *
EMPLIB $8 MADE SOURCE OUTPUT FILE NEW *
$SKIP 2
EMPLIB $8 SKIPPED 2 FILES ON AGO *
$SHOREWIN
SCHANGER PROG VERS
EMPLIB $8 CHANGED 9TH FILE (BINARY PROG VERS 10/29/71 , IS NOW BINARY PROG VERS 1/29/71 )
ON EMPLIB USING CONTENTS OF AGO FILE.
$TOC
EMPLIB $8 TABLE OF CONTENTS OF EMPLIB
1 EMPLIB
2 TOC
3 PROG VERS
4 PROG VERS
$RENAME PROG VERS, PROGA NEWNAME
$ ADD NEWPROG
EMPLIB $8 ADDED 9TH FILE (SOURCE NEWPROG 10/29/71 ) TO EMPLIB FROM OLD FILE.
$TOC
EMPLIB $8 TABLE OF CONTENTS OF EMPLIB
1 EMPLIB
2 TOC
3 PROGA NEWNAME
4 PROG VERS
5 NEWPROG
$DROP NEWPROG
EMPLIB $8 DROPPED 9TH FILE (SOURCE NEWPROG 10/29/71 ) FROM EMPLIB.
$ TOC
EMPLIB $8 TABLE OF CONTENTS OF EMPLIB
1 EMPLIB
2 TOC
3 PROGA NEWNAME
4 PROG VERS
5 SOURCE
4  PROG  VERS  10/29/71  BINARY

$KEEP  PROG  VERS
EMPLIB $S$ KEEP 4TH FILE (BINARY PROG  VERS  10/29/71 ) FROM EMPLIB ON ARCHIV FILE.
1 SOURCE AND BINARY FILES NOW KEPT ON ARCHIV FILE.

$KEEP  PROGA  NEWNAME
EMPLIB $S$ KEEP 5TH FILE (SOURCE PROGA  NEWNAME  10/29/71 ) FROM EMPLIB ON ARCHIV FILE.
2 SOURCE AND BINARY FILES NOW KEPT ON ARCHIV FILE.

$HISTORY
EMPLIB $S$ HISTORY OF ARCHIV
KEEP NO.  1  PROG  VERS  10/29/71  BINARY
KEEP NO.  2  PROGA  NEWNAME  10/29/71  SOURCE

$RUN  PROG
EMPLIB $S$ COPIED 4TH FILE (BINARY PROG  VERS  10/29/71 ) FROM EMPLIB TO XOT FILE.

$COPY  PROGA  NEWNAME
EMPLIB $S$ COPIED 5TH FILE (SOURCE PROGA  NEWNAME  10/29/71 ) FROM EMPLIB TO NEW FILE.

$SELECT  PROG  VERS,151,SPLITR, SPLITC
EMPLIB $S$ COPYING THE FOLLOWING BINARY RECORDS ONTO XOT FROM THE 4TH FILE (BINARY PROG  VERS  10/29/71 ) ON EMPLIB.

SELECTED  REFUSED
150  PLOT
151  LLL
SPLITR
SPLITC
QRFIT
FINDER
LODVAL
REDUC
RITETS
ASTRAN
ORPAN
RITRAN
REDON
BLANKS
EXON
LINES

END OF COPY

$REFUSE  PROG  VERS,150,SPLITR,SPLITC
EMPLIB $S$ COPYING THE FOLLOWING BINARY RECORDS ONTO XOT FROM THE 4TH FILE (BINARY PROG  VERS  10/29/71 ) ON EMPLIB.

SELECTED  REFUSED
150  PLOT
151  LLL
SPLITR
SPLITC
QRFIT
FINDER
LODVAL
REDUC
RITETS
ASTRAN
ORPAN
RITRAN
REDON
BLANKS
EXON


Appendix B. PROGRAM LISTING
PROGRAM EPLIB

COMMON //LMAX, MAX(6000)
COMMON //ARGS/NAME, NAMS, NAMOLD, IVOLD, REG, NAME(100), JTIC, LASTF,
* ARCH
DIMENSION CARD(60), TOC(4,50), MODE(2)
COMMON //FILES/FILNAM(9), FEATS(9), XI(1)

INTEGER TOC, ODATE, A, CHAR, DOL, CARD, FILNAM
DATA CHAR(29), TOC, ODATE, A, CHAR, DOL, CARD, FILNAM

FORMAT(*EPLIB IS THE DATE IS A10, **AND THE WORK FILES ARE*)
IFLAG = 0
READ 2, CARD
FORMAT(*EPLIB IS INVALID CONTROL CARD FOLLOWS, JOB WILL BE ABOR-
'TED AFTER READING INPUT FILE.*)
IFLAG = 1
READ 2, CARD
FORMAT(1048)
IF(CARD(1).EQ.0) GO TO 1000
IF(CARD(1).EQ.0) GO TO 20
PRINT 3
3 FORMAT(*EPLIB IS INVALID CONTROL CARD FOLLOWS, JOB WILL BE ABOR-
'TED AFTER READING INPUT FILE.*)
IFLAG = 1
PRINT 4, CARD
4 FORMAT(1048)
IF(IFLAG.NE.0) GO TO 10
IF(IFLAG.NE.0) GO TO 2500
I = 1
CALL NEXTWO(CARD(2), I, J, K)
IF(K.NE.0 OR J.NE.CHAR(21)) GO TO 2500
IFIRST = 1
CONTINUE
PROGRAM EMPLIB

GO TO 30

2500 CONTINUE
IF(JTOC.NE.0) GO TO 30
CALL GETTOC(TOC,NFILES,JCR)

60IF(JCR.EQ.1 AND .NOT.JTOC.EQ.0) PRINT 17,FILNAM(*.
17 FORMAT(* EMPLIB $$$$ TOC MISSING ON *,A7,*,*.)
JTOC=1
IF(JCR.EQ.1) JTOC=-1
REWINd 4

65 LASTF=0
10 CALL ISI1(CHAR,CARD(2),NCHAR,JUMP,IFILE,TOC,DDATE,JCR,FILNAM)
IF(JUMP.NE.0) GO TO 40
PRINT 19
19 FORMAT(* EMPLIB $$$$ WILL ABORT AFTER READING INPUT FILE.*)
IFLAG=1
GO TO 10
40 CONTINUE
GO TO (60,60,60,70,70,80,80,90,90,100,100,110,110,110,110,150,150,160,160,210,230,
JUMP 70
70 CONTINUE

60 CONTINUE
CALL POSFIL(4,LASTF,IFILE)

80 CONTINUE

90 CONTINUE
CALL CVICK0I
IF(JUMP.EQ.2) K=2
CALL CPYFIL(*,K,0)

83 LASTF=IFILE
J=TOC(4,IFILE)
PRINT 6,IFILE,MODE(J),TOCII,IFILE,J=1,J3,FILNAM(J),FILNAM(K)
6 FORMAT(* EMPLIB,49A9,* FILE (*,A7,3A10,* FROM *,A7,* TO *,A7,* FILE.*)
GO TO 10
77 CONTINUE
C CHANGE AND CHANGED

92 CONTINUE

IVOLD=TOC(2,IFILE)
75 CONTINUE

OLDATE=TOC(3,IFILE)
TOC(3,IFILE)=ODATE
REWINd 4
95 REWINd 40

70 CONTINUE
CALL CPYFIL(4,40,NFILES)
J=9
IF(JUMP.EQ.5 OR JUMP.EQ.29) J=3
IF(NOREW.EQ.0) REWINd J
133 NOREW=0
CALL NEWFIL(NFILES,TOC,IFILE,J)

77 CONTINUE

105 CONTINUE
LASTF=0
K=TOC(4,IFILE)
PRINT 7,IFILE,MODE(K),IVOLD,OLDATE,MODE(K),K

135 TOCII,IFILE,J=1,J3,FILNAM(J),FILNAM(I)
7 FORMAT(* EMPLIB $$$$ *,A9,* FILE (WAS *,A7,3A10,* IS NOW *
* *,A7,3A10,* )/20K,* ON *,A7,* USING CONTENTS OF *,A7,* FILE.*)
GO TO 10
80 CONTINUE

110 C ADD AND ADD
PROGRAM EMLIB

COC 6600 FTN V3.0-P29Z OPT=1 04

J=8
IF(JUMP.EQ.7) J=3
REWIND 4
REWIND 40
IF(NOREM.EQ.3) READING J

CALL CPYFIL(4,40,NFILES)
NFILES=NFILES
CALL NEOUT(NFILES,TOC,NFILES,J)

LASTF=0
K=TOC(4,JFILE)
PRINT 6,IPYR4ILJ,TOC(J,JFILE),NFILES,J
FORMAT(" EMLIB $$$ ADDED *",I2,"TH FILE (*.A7,3A10,*) TO *.A7,* FR"
OM *,A7,* FILE.*)

GO TO 10
90 CONTINUE
C DROP AND DROPB
REWIND 4
REWIND 40

CALL CPYFIL(4,40,NFILES)
J=TOC(4,JFILE)
PRINT 9,JFILE,MODE(J),TOC(I,JFILE),I=1,3,FILNAM(I),FILNAM(J)
J=TOC(4,JFILE)
TOC(J,JFILE)-1
GO TO 96
DO 95 I=1,4
TOC(J,1)=TOC(K,I+1)
95 CONTINUE
96 CONTINUE
C CALL NEOUT(NFILES,TOC,JFILE,0)
LASTF=0
NFILES=NFILES-1

FORMAT(" EMLIB $$$ DROPPED *",I2,"TH FILE (*.A7,3A10,*) FROM *.A7,
*,*)
GO TO 10
100 CONTINUE
C TOC

PRINT 11,FILNAM(4)
FORMAT(" EMLIB $$$ TABLE OF CONTENTS OF *.A7")
DO 105 I=1,NFILES
IF(I.EQ.2) GO TO 104
K=TOC(4,I)
PRINT 31,I,TOC(J,1),J=1,3,MODE(K)
105 CONTINUE
104 CONTINUE
PRINT 32,TOC(J,1),J=1,4)
FORMAT("0X,310X,410X,10*
FILES ON LIBRARY*)
105 CONTINUE
GO TO 10
C KEEP AND KEEPB
110 K=FETS(5)
J=2*IARCH
IF(IARCH.NE.0) GO TO 120

165
PROGRAM EMLIB

I=0
REWIND 5
BUFFER IN(5,1) (A,A(2))
I=I+1
170 IFUNIT(5) 130,125,2000
125 I=0
J=J+1
GO TO 120
130 IF(LENGTH(5).NE.1) GO TO 120
IF(A(1).NE.LASTH) GO TO 120
IARCH=J/2
140 CALL SKIPS(K(K),1)
CALL POSFIL(4,LASTF,IFILE)
BUFFER OUT(5,1) (TOC(4,IFILE),TOC(4,IFILE))
180 IFUNIT(5).GE.0) GO TO 2010
ENDFILE 5
CALL CPYFIL(4,5,1)
LASTF=IFILE
BUFFER OUT(5,1) (LASTH,LASTH)
145 IFUNIT(5).GE.0) GO TO 2010
BACKSPACE 5
IARCH=IARCH+1
I=IARCH
J=TOC(4,IFILE)
190 PRINT 12,IFILE,MODE(J),IFILE,FILNAM(4),FILNAM(5),I,
'FILNAME(5)
12 FORMAT* EMLIB $$ KEPT *,-2,*,TH FILE (*.A7,3A10,*) FROM *,-A7,* G
* N =,-A7,* FILE.=I20,* SOURCE AND BINARY FILES NOW KEPT ON *,-A7,* F
*ILE.*
195 GO TO 10
150 CONTINUE
C CREATE
REWIND 4
CALL CPYFIL(4,40,1)
CALL NENFIL(1,TOC,0,0)
REWIND 4
LASTF=0
PRINT 16,FILNAME(4)
16 FORMAT* EMLIB $$ CREATED EMLIB ON FILE NAMED *,-A7,*.
JTOC=0
GO TO 10
160 CONTINUE
C FIND AND FINDB
IARCH=0
REWIND 5
I=0
N=1
IF(JUMP.EQ.15) M=2
J=1
170 BUFFER IN(5,1) (A,A(4))
I=I+1
IFUNIT(5) 190,160,2020
180 I=0
J=J+1
GO TO 170
PROGRAM EMPLIB

190 L=LENGTH(5)
191 IF(A(1).NE.LASTH.OR.L.NE.1) GO TO 200
192 PRINT C1,MODE(m),NAME,IVERS,FILNAM(5)
193 FORMAT(* EMPLIB *** FILE SOUGHT (*,A7,2A10,*), NOT ON *,A7,*,*)
194 IFLAG=1
195 GO TO 10
200 IF(L.NE.4) GO TO 170
201 IF(M,NAME,NE.A(1),NE.IVERs,NE.A(2)) GO TO 170
202 I=I+2
211 PRINT 221,MODE(m),A(1),A(2),A(3),FILNAM(5)
212 FORMAT(* EMPLIB *** 1,12,12TH FILE FOUND (*,A7,3A10,*), ON *,A7,*,*)
213 BUFFER IN(5,1) (A,A)
214 IF(UNIT(5).NE.0) GO TO 2020
215 M=I-M
216 CALL CPYFIL(5,0,0)
217 PRINT 23,FILNAM(5)
218 FORMAT(* EMPLIB *** COPIED FILE FOUND TO *,A7,*,*)
219 IARCH=I
220 GO TO 10
221 CONTINUE

C HISTORY

240 PRINT 25,FILNAM(5)
241 FORMAT(* EMPLIB *** HISTORY OF *,A7)
242 I=0
243 220 I=I+1
244 IARCH=I-1
245 BUFFER IN(5,1) (A,A(4))
246 IF(UNIT(5).GE.0) GO TO 2030
247 IF(LENGTH(5).NE.1.OR.A(1).NE.LASTH) GO TO 225
248 BACKSPACE 5
249 GO TO 10
250 CONTINUE

255 K=A(4)
256 PRINT 26,I,J,MODE(K)
257 FORMAT(21X,KEEF 40.,1I4,10X,4A10)
258 BUFFER IN(5,1) (A,A)
259 IF(UNIT(5).NE.0) GO TO 2030
260 CALL CPYFIL(5,0,1)
261 GO TO 220
262 CONTINUE

C RENAME AND RENAMED

265 REMIND 40
266 CALL CPYFIL(4,40,NFILES)
267 CALL NEWFIL(NFILES,T0C(0,0)
268 LASTF=0
269 K=T0C(4,IFILE)
270 PRINT 28,IFILE,MODE(K),NAMOLD,IVOLD,T0C(3,IFILE),MODE(K),TOC(1,IF
271 FILE),T0C(1,3)
272 FORMAT(* EMPLIB *** RENAMED *.*,12,12TH FILE WAS *,A7,3A10,*), IS NOW
273 *,A7,3A10,*,*)
274 GO TO 10
275 CONTINUE

C REFUSE AND SELECT

28
PROGRAM EMPLIB

IRS=1
IF(IFILE.EQ.0) GO TO 245
CALL POSFIL(*,LASTF,IFILE)
K=TOC(*,IFILE)

280 PRINT 25,FILNAM(1),IFILE,MODE(K),TOC(*,IFILE),I=1,3,FILNAM(4)
FORMAT(*$ EMPLIB $$ COPYING THE FOLLOWING BINARY RECORDS ONTO *.,AT

290 I=12,*TH FILE (*.,AT,I=10,*$ ON *.,AT,*,$/**/AT,*$SELECTED

285 $*,*REFUSED$)
K=NREC
IF(JUMP.EQ.20) K=-K
CALL CPYREC(*,NAMRECK)
PRINT *FILNAM(1),IFILE,LASTF=IFILE
GO TO 10

245 CONTINUE

290 PRINT 36,FILNAM(1),FILNAM(3)
FORMAT(*$ EMPLIB $$ COPYING THE FOLLOWING BINARY RECORDS ONTO *.,AT

360 I=12,*TH FILE (*.,AT,I=10,*$SELECTED$*,*REFUSED$*

295 $*,*REFUSED$)
K=NREC
IF(JUMP.EQ.20) K=-K
CALL CPYREC(*,NAMRECK)
GO TO 10

250 CONTINUE
C NOWREWIND
NOREM=1
GO TO 10

260 CONTINUE
C SKIP AND SKIPO
I=0
IF(JUMP.EQ.26) I=3
CALL CPYFIL(*,I,IFILE)
PRINT 33,FILNAM(1)

330 FORMAT(*$ EMPLIB $$ SKIPPED*$I*,* FILES ON *.,AT,*$)

270 CONTINUE
C CREATEARCH
REWIND 5
IARCH=0
A(I)=NHLAST
BUFF=OUT(5,1) (A,A)

315 IF(UNIT(5).GE.0.0) GO TO 2010
REWIND 5
PRINT 35,FILNAM(5)

350 FORMAT(*$ EMPLIB $$ CREATED ACHIVE ON FILE NAMED *.,AT,*$)
GO TO 10

280 CONTINUE
C REPLACE AND REPLACED
L=8REPLACED
GO TO 75

320 CONTINUE
H=LMAX+1
IF(IRS.NE.0) PRINT 34,MX,M

340 FORMAT(*$ EMPLIB $$ MAXIMUM RECORD LENGTH PROCESSED FOR SELECT-REF


350 USE MAX,*$,$ IS THE MAXIMUM ALLOWED.$*$
IF(IFLAG.EQ.0) PRINT 13

330 13 FORMAT(*$ EMPLIB $$ FINISHED $$*)
PROGRAM EMPLIB

REWIND 4
DO 1010 J=1,2
  IF( (I) .AND. 500) .NE. 20) GO TO 1010
  END FILE J
  CALL SKIP (I, 0)
1010 CONTINUE
  IF( IFLAG .EQ. 0) STOP
  PRINT 10
  FORMAT('**EMPLIB $$$ ABORTING $$$')
  CALL ABORT
2000 CONTINUE
  PRINT '14, I, J, FILNAM(5)
  14 FORMAT('** EMPLIB $$$ KEEP READ PARITY ERROR ON*5, *TH RECORD ON*, I
    *5,*TH FILE ON *, A7, *')
  CALL ABORT
2010 CONTINUE
  PRINT '15, FILNAM(5)
  15 FORMAT('** EMPLIB $$$ KEEP WRITE PARITY ERROR ON *, A7, *')
  CALL ABORT
2020 CONTINUE
  PRINT '24, FILNAM(5)
  24 FORMAT('** EMPLIB $$$ FIND READ ERROR ON *, A7, *')
  CALL ABORT
2030 CONTINUE
  PRINT '27, FILNAM(5)
  27 FORMAT('** EMPLIB $$$ HISTORY READ ERROR ON *, A7, *')
  CALL ABORT
END
SUBROUTINE GETTOC

SUBROUTINE GETTOC(IOC, NFILES, JCR)
DIMENSION TOC(4,50)
REWIND 4
JCR=0
CALL CPYFIL(4,0,1)
BUFFER 1h(4,1) (TOC,TOC(4,50))
IF(UNIT(4)) 10,20,100
10 NFILES=LENGTH(4)/4
RETURN
20 NFILES=2
JCR=1
RETURN
100 PRINT 1
FORMAT(* EMPLIG $S$ GETTOC PARITY ERROR*)
15 CALL ABORT
END
SUBROUTINE CPYFIL

SUBROUTINE CPYFIL(IN,NF)
COMMON /LP15,*(1)
COMMON /FILLS,FILNAMG,FETS(9),X(1)
INTEGER FILLAP,FETS,X
IF(IN.GT.0) GO TO 30
DO 20 I=1,NF
10 BUFFER IN(IN),*,X
IF(IN.NE.IN) 10,20,200
20 CONTINUE
RETURN
30 CONTINUE
LMS=512*(LMAX/L12)
JIN=FILNAM(IN)
JOUT=FILNAM(IN)
J=FETS(JIN)
K=FETS(JOUT)
DO 40 I=1,NF
20 IF(I.EQ.0) GO TO 36
IF(I.EQ.1) GO TO 400
PRINT 1,FILNAM(JIN)
3 FORMAT(* EMPLIE $$**,A7,** INITIALLY PCSITICNEC AT END-OF-INFORNAT
*IOM, EMPLIE ANCHITIN.*
25 CALL APORT
36 CONTINUE
IF(I.EQ.1) GO TO 300
IF(IN.GT.0) FNGFIL JOUT
40 CONTINUE
30 RETURN
200 CONTINUE
PRINT 1,FILNAM(JIN)
1 FORMAT(* EMPLIE $$ CPYFIL READ PARITY ERROR IN**,I3,**TH FILE (FROM
* START OF CPYF) ON **,A7,***)
35 CALL ACR
300 CONTINUE
PRINT 6,FILNAM(JIN),FILNAM(JOUT),I,IER
2 FORMAT(* EMPLIE $$ I/O ERROR IN CPYBUF WHILE COPYING **,A8,**TC **,A
*7,**, FILE NUMBER**I3/20X,**ERROR CODE IN CCTAL IS **,C20)
40 CALL ACR
400 CONTINUE
PRINT 5,FILNAM(JIN),I,NF
5 FORMAT(* EMPLIE $$ END-OF-INFORMATION ENCOUNCTING COPYING**,I3,**TH
* OF**,I3,** FILES FROM START OF CPY) ON **,A7,***)
45 CALL ACR
END
SUBROUTINE GETFIL

COMMON /FILES/FILNAM(9),FETS(9),X(1)
INTEGER FILNAM,FETS,X
DATA MASK//7777777/

L=LOCF(X)
DO 10 I=1,9
 J=I+1
 FILNAM(I)=X(J)
 J=FILNAM(),AND.MASK
 FETS(I)=J=L+1
10 CONTINUE
RETURN
END
SUBROUTINE SWAPFIL(IUNIT,NAME)

COMMON /FILES/FILNAM(9),FETS(9),X(1)

DATA WORD1,WORD2,IOBINARY INP,10HEMPLIB AR
*,10NARCHIVE KE,10HEMPLIB OUT,10HEMPLIB OUT,10HEMPLIB OUT,10NARCHIVE INP,10NARCHIVE INP,
*PUT,3MPUT,3MPUT,3MPUT,3MPUT,3MPUT,3MPUT,3MPUT /

I=0
IF((II. AN7O."ASKDT).EQO.(X(I.L)).AND. (X(I.L). AND. NOT.MASKC).EQO) GO TO 10
KFLAG=1

C CHECK TO SEE IF LAST OPERATION WAS AN FOR WRITE.
IF(IN.EQ.28) GO TO 20
ENDIF IUNIT
CALL SKIP3X(IUNIT,0)

CONTINUE
X(I+1)=X(I+1). AND. NOT.MASKCT
X(I+4)=X(I+4). AND. MASK

CONTINUE
IN=NAME
DO 20 J=4,10
M=SHIFT(MASKCH,6*(J-1))
IF((IN. AND. M).NE.(IB. AND. M)) GO TO 30
IN=IN. AND. NOT.M
20 CONTINUE

CONTINUE
X(I+1)=X(I+1).AND. NOT.MASKC.OR.3
I=FILNAN(IUNIT)
FILNAM(IUNIT)=IN
IF(KFLAG.GT.0) GO TO 40

RETURN
40 PRINT 1,WORD1(IUNIT),WORD2(IUNIT),FILNAM(IUNIT)
RETURN
END

10 I=FETS(IUNIT)
II=X(I+1)
KFLAG=0
IF((II. AN7O.MASKCT).EQO.(X(I+1)).AND. (NOT.MASK).EQO) GO TO 10
KFLAG=1

10 CONTINUE
I=FETS(IUNIT)
II=X(I+1)
KFLAG=0
IF((II. AN7O.MASKCT).EQO.(X(I+1)).AND. (NOT.MASK).EQO) GO TO 10
KFLAG=1

20 CONTINUE
X(I)=X(I). AND. NOT.MASKC
X(I+1)=X(I+1). AND. NOT.MASKC

30 CONTINUE
X(I)=X(I). AND. NOT.MASKC OR.3
I=FILNAN(IUNIT)
FILNAM(IUNIT)=IN
IF(KFLAG.GT.0) GO TO 40

RETURN
40 PRINT 1,WORD1(IUNIT),WORD2(IUNIT),FILNAM(IUNIT)
RETURN
END
CCMPASS - VER 2.  01/18/72  11.12.49.

IDENT SKIPB
ENTRY SKIPB
EXIT CPC
VFD 30/5H/SKIPB, 30/1
SKIPB BSS 1
SIX A0
SAP SAVAO
SA2 A1+1
SA1 X2
S1 X1
NZ X3, SKIPFIL
RJ CPC
VFD 10/3, 2/1, 22/1, 10/6400
EQ RET
SKIPFIL SA4 ARG
LX3 10
BX6 X3+X4
SA6 ARGLOG
RJ CPC
ARGLOC BSS 1
RET SA5 SAVAO
SA6 X5
EQ SKIPO
SAVAO BSS 1
ARG VFD 10/3, 2/1, 22/0, 4/178, 14/6400
END

STORAGE USED  26 STATEMENTS  7 SYMBOLS
6600 ASSEMBLY  0.121 SECONDS  16 REFERENCES
CCPASS - VER 2. 01/18/72 11:12:50.

IDENT CPYBUF
ENTRY CPYBUF
WFO 35/GHCOPYBUF.2*/5

CPYBUF
SSS 1
SBF 1
SAC 0*07
SA2 AI+07
SA3 A1+06
SA4 A2+06
SA5 A3+06
SA6 A0
SA6 AZER0
SP X5
SA7 IER
MXF 0
SBF X5
SA7 FLAG
BK6 X1
SA6 BOUNDS
BK7 X2
SA7 A6+07
SA1 X3+07
SA2 AI+O7
BK6 X1
BK7 X2
SA1 A2+07
SA2 AI+07
SA6 SAVE
SA7 A6+07
BK6 X1
BK7 X2
SA6 A7+07
SA7 A6+07
SA1 BOUNDS
SR2 X1
SA2 AI+07
SD1 X2
SA1 X3+07
MXO 42
SA7 02
BK6 X1*X0
BK6 X6*X7
SA6 X3+07
SA7 A6+07
SA7 A7+07

X3 TO CONTAIN ACCESS OF FILEIN
X4 SAME FOR FILEOUT

LOOP
36
SA1 A1+3
SX2 01
BX6 XI*X0
BX7 X6*X2
SA7 A7+87
SX1 120
SA2 X3
BX6 X0*X2
BX7 X6*X1
SA7 X3
SA1 CIOMORD
BX6 X1*X3
SA6 07
RECALL SA5 67
NZ X5, RFCALL
SA1 X3
SX0 3F0000
BX2 X1*X0
ZR X2, DK
SX0 30000
BX2 X1*X0
NZ X2, EOI
SA5 IE1R
BX6 X1
SA6 X5
EQ RETURN
FOI SX2 7400338
MX0 42
BX6 X0*X1
BX6 X6*X2
SA6 X3
SA2 FLAG
NZ X2, RETURN
SX6 67
SA2 IE1R
SA6 X2
EQ RETURN
CK SX0 77B
BX7 X0*X1
SX0 33B
IX6 X7-X0
ZR X6, RETURN
SA1 X3+87
SA2 X4+87
HX0 42
BX6 -X0*X1
BX7 X0*X2
BX7 X6*X7
SA7 X2
SA1 A1+87
BX6 X1
SA6 A7+87
SA2 A1+87
BX7 X2
SA7 A6+87
SA1 A2+87
SA2 A7+87
GCOMPASS - VER 2.  01/10/72  11.12.50.

BX6 -X0*2
BX7 X0+X2
BKX X6+X7
SA7 A2
SX6 87
SA7 FLAG
SA1 X3
SA2 X4
SA3 X1
SX6 B3+3
BKX X6*X2
BKX X6+X7
SA7 X4
SA1 CIOM WORD
DX6 X1*X4
SA6 B7
RECALLA SA5 B7
M2 X5,RECALLA
SA1 X6
S10 3F000B
BX2 X1*X0
ZR X2,OKA
S19 IER
BX6 X1
SA6 X5
ED RETURN
OKA SB1 -378
SB2 X1+87
NE B2,LOOP
RETURN SA1 SAVE
SA2 A1+87
M8 42
SA5 X3+87
BX6 -X0*X1
BX5 X0*X5
BX6 X6+X5
BKX X2
SA1 A2+87
SA2 A1+87
SA5 X3+4
SA6 X3+97
SA7 A5+87
BX6 X1
BKX -X0*X2
BX5 X0*X5
BKX X5+X5
SA6 AP+87
SA7 A6+87
SA1 A2+87
SA2 A1+87
SA5 X4+87/
BX6 -X0*X1
BX5 X0*X5
BX6 X6+X5
BKX X2
SA1 A2+87
SA2 A1+87
COMPASS - VER 2. 01/10/72 11.12.50.

SAS X4+4
SAS X4+07
SAT A6+07
BX6 X1
BX7 -X0*X2
BX5 X0*X5
BX7 X7+X5
SA6 A7+07
SAT A6+07
SAI AZERO
SAI X1
ED CPYBUF
IER BSS 1
AZERO BSS 1
BOUNDS BSS 2
FLAG BSS 1
SAVE BSS 8
CIWORD VFO B/INCIO,2/1,40/0
END

STORAGE USED 190 STATEMENTS 14 SYMBOLS
6600 ASSEMBLY 0.536 SECONDS 42 REFERENCES
SUBROUTINE NEWFIL(NFILES,TOC,IFILE,J)
DIMENSION TOC(4,NFILES)
REWIND 4
REWIND 40
L=MAX0(2,NFILES)
K=L
IF(IFILE.NE.0.AND.J.EQ.0) K=K-1
GO 30 I=1,L
IF(I.NE.2) GO TO 10
BUFFER OUT(4,3) (TOC,TOC(4,K))
IF(UNIT(4).GE.0) GO TO 100
ENDIF 4
IF(NFILES.NE.1) CALL CPYFIL(40,0,1)
GO TO 30
IF(J.E.0) GO TO 20
IF(NFILES.NE.IFILE) CALL CPYFIL(40,0,1)
IF(J.NE.0) CALL CPYFIL(J,4,1)
GO TO 30
20 CALL CPYFIL(40,4,1)
30 CONTINUE
REWIND 4
RETURN
100 CONTINUE
PRINT 1
FORMAT(* EMPLIB $$$ TOC WRITE PARITY ERROR IN NEWFIL. *)
CALL ABORT
END
SUBROUTINE POSFIL

SUBROUTINE POSFIL(N,LASTF,IFILE)
COMMON /FILES/FILNAME(9),FETS(9),X(1)
INTEGER FILNAME,FETS,X
IF(IFILE.LT.LASTF) GO TO 10
5 I=FETS(N)
CALL SKIPB(X(I),LASTF-IFILE+1)
GO TO 30
10 JFILES=IFILE-LASTF-1
IF(JFILES.EQ.0) GO TO 30
CALL CPYFILE(*0,JFILES)
30 LASTF=IFILE-1
RETURN
END
SUBROUTINE ISIT

SUBROUTINE ISIT(CHARC, NAME, IVERS, NAMOLO, IVOLD, NREC, NAHREC(100), JTOC, LASTF, ICARCH)

INTEGER FILTYP(8), NTYP(8)

INTEGER FILNAM(1)

DIMENSION CARO(79), TOC(4, 1), CHAR(1)

DIMENSION NUMS(10)

INTEGER CARD, TOC, DATE, CHAR

INT yPE FILTYP/2HeO, 2H01, 2H50, 2H51, 1H4L, 1H4H, 1H4Q, 1H4F, NTYP/1, 3-2, 8, 4, 5,

,7, 6/

DATA FILTYP/2HeO, 2H01, 2H50, 2H51, 1H4L, 1H4H, 1H4Q, 1H4F, NTYP/1, 3-2, 8, 4, 5,

,7, 6/

DATA I/I±H

#MASK/77B/

DATA MOOEH/6HSOURCE, 6HDINARY/

DATA NUMS/IH2, IH4, IH5, IH6, IH7, IH8, IH9/

JUMP=0

NAME=10

IST=1

CALL NEXTHD(CARD, IST, NAME, JFLAG)

IF(JFLAG.EQ.0) GO TO 10

CONTINUE

IF(CCHARCJUMP).EQ.111 IF) GO TO 50

CONTINUE

PRINT 1, CARD

1 FORMAT(* EMPI, ** $ IMPROPER DIRECTIVE ON CARD. **, 79A1)

CONTINUE

GO 40 JUMP=1, NCHAR

IF(CCHARCJUMP).EQ.2, NAME) GO TO 50

CONTINUE

PRINT 1, CARD

2 FORMAT(* EMPI, ** $ UNRECOGNIZABLE DIRECTIVE. **, A10, **, 79A1)

JUMP=0

RETURN

CONTINUE

IF(JUMP.EQ.27) RETURN

IF(JUMP.EQ.13) GO TO 300

IF(JUMP.EQ.21) GO TO 500

IF(JUMP.EQ.0) GO TO 55

IF(JUMP.EQ.13.AND.JUMP.NE.17.AND.JUMP.NE.19) GO TO 55

JUMP=0

PRINT 7, NAME

7 FORMAT(* EMPLIB $ DIRECTIVE REQUIRES TABLE OF CONTENTS, WHICH HAS NOT BEEN CREATED. **, A10)

RETURN

CONTINUE

IF(JUMP.EQ.10.OR.JUMP.EQ.11) RETURN

IF(JUMP.EQ.24) RETURN

IF(JUMP.EQ.25.OR.JUMP.EQ.26) GO TO 400

IF(JUMP.EQ.22.OR.JUMP.EQ.23) GO TO 700

NAME=18

CALL NEXTHD(CARD, IST, NAME, JFLAG)

IF(JFLAG.EQ.111 OR.JUMP.EQ.19.AND.JUMP.EQ.20)) GO TO 56

IFILE=0

GO TO 630

CONTINUE

IF(JFLAG) 70, 80, 60

CONTINUE

PRINT 3, NAME, CARD

3 FORMAT(* EMPLIB ** $ CANNOT FI: D PROGRAM NAME ON CARD. **, 79A1)

JUMP=3

CONTINUE

IFILE=0

GO TO 630

CONTINUE
SUBROUTINE ISIT  CDC 6600 FTN V3.0-P292 OPT=1  W4

RETURN
70 CONTINUE
PRINT 4,NAME,CARD
4 FORMAT(* EMPLIB $S$ PROGRAM NAME TOO LONG.*$A10,**$A79A1)
JUMP=0
RETURN
60 CONTINUE
CALL NEXTWO(CARD,IST,IVERS,JFLAG)
65 IF(JFLAG.GE.0) GO TO 100
PRINT 5,CARD
5 FORMAT(* EMPLIB $S$ VERSION NAME TOO LONG.*$A10,**$A79A1)
JUMP=0
RETURN
70 100 CONTINUE
IF(JFLAG.EQ.1) IST=IST-1
IF(JUMP.EQ.14.OR.JUMP.EQ.15) RETURN
N=TOC(4,2)
MODE=1
75 IF(JUMP.EQ.29) MODE=2
IF(JUMP.EQ.3.OR.JUMP.EQ.5.OR.JUMP.EQ.9) MODE=2
IF(JUMP.EQ.1.OR.JUMP.EQ.7.OR.JUMP.EQ.12.OR.(JUMP.GE.18.AND.JUMP.LE
*.20)) MODE=2
DO 160 IFILE=1,N
80 IF(TOC(4,IFILE).EQ.MODE.AND.TOC(1,IFILE).EQ.NAME.AND.TOC(2,IFILE).EQ.
'INVALID,NEAME,GO TO 170
IF(JUMP.EQ.1.OR.TOC(4,IFILE).NE.MODE.OR.TOC(1,IFILE).NE.NAME) GO T
O 160
90 C RUN
IVERS=TOC(2,IFILE)
RETURN
160 CONTINUE
IF(JUMP.EQ.6.OR.JUMP.EQ.7) GO TO 180
PRINT 6,NAME,IVERS
6 FORMAT(* EMPLIB $S$ NOT IN TOC.$A)
JUMP=0
RETURN
170 CONTINUE
IF(JUMP.EQ.6.OR.JUMP.EQ.7) GO TO 185
95 C RUN, COPY, KEEP, AND KEEPS
IF(JUMP.EQ.1.OR.JUMP.EQ.2.OR.JUMP.EQ.3.OR.JUMP.EQ.11.OR.JUMP.EQ.12
* RETURN
IF(JUMP.NE.4.AND.JUMP.NE.5) GO TO 190
C CHANGE AND CHANGES
RETURN
100 CONTINUE
C ADD AND A309
IFILE=TOC(4,2)+1
TOC(4,2)=IFILE
TOC(1,IFILE)=NAME
TOC(2,IFILE)=IVERS
TOC(3,IFILE)=DATE
TOC(4,IFILE)=JUMP-5
RETURN
105 185 CONTINUE
SUBROUTINE ISIT

J=TOC(4,IFILE)
PRINT 11,IFILE,MODEM(I),TOC(I,IFILE),I=1,3
11 FORMAT(*,EMPLIB ** ADDING FILE ALREADY IN TOC IS NOT PERMITTED. ** FILE IS *,I2,** FILE (*,A7,3X,10,A5,**)
JUMP=0
RETURN
190 IF(JUMP.NE.9.AND.JUMP.NE.9) GO TO 210
C DROP AND DROP
IF(IFILE.GT.2) RETURN
PRINT 5
FORMAT(*,EMPLIB ** DROPPING LIBRARY OR TABLE OF CONTENTS IS NOT ** PERMITTED.*)
JUMP=0
RETURN
125 IF(JUMP.NE.17.AND.JUMP.NE.18) GO TO 600
C RENAME AND RENAMED
CALL NEXTWDICARD,IST,NAME,JFLAG
IF(JFLAG.EQ.1) GO TO 220
PRINT 12,CARD
12 FORMAT(*,EMPLIB ** MISSING COMMENT.),A10,A)
JUMP=0
RETURN
220 IF(JFLAG.GE.0) GO TO 260
215 PRINT 12,CARD
PRINT 3,CARD
JUMP:0
RETURN
250 IVERS=10
CALL NEXTWDICARD,IST,IVERS,JFLAG
IF(JFLAG.GE.0) GO TO 260
230 PRINT 3,CARD
PRINT 4,NAME,CARD
JUMP=0
RETURN
240 PRINT 5,J,NAME,CARD
JUMP=0
RETURN
255 PRINT 4,CARD
PRINT 4,IVERS,CARD
JUMP=0
RETURN
260 CONTINUE
RETURN
153 IVOLD=TOC(2,IFILE)
TOC(1,IFILE)=NAME
TOC(2,IFILE)=IVERS
RETURN
265 CONTINUE
C CREATE
TOC(1,2)=3HTOC
TOC(2,2)=10
TOC(3,2)=ODATE
TOC(4,2)=2
TOC(1,1)=6EMPLIB
TOC(2,1)=IB
TOC(3,1)=ODATE
TOC(4,1)=2
JCR=0
RETURN
SUBLROUTINE ISIT

C FILESA
500 CONTINUE
C FILES
510 CALL NEXTWO(CARD,IST,NAM£JFLAG)
 IF(JFLAG.EQ.2) RETURN
 IF(JFLAG) 520,530,510
520 PRINT 13,NAM£CARD
13 FORMAT(EMPLIB $S$ WORD IS TOO LONG..A10,***79A1)
 JUMP=0
 RETURN
170 IF(JFLAG) 520,530,510
530 CONTINUE
 DO 540 I=1,8
 IF(NAME.EQ.520) GO TO 550
540 CONTINUE
 PRINT 14,NAM£FILYP
14 FORMAT(# EMPLIB $S$ FILE TYPE *A10,* IS NOT ONE OF THE ALLOWED FO
 'RMS *,S9A3,* *)
 JUMP=0
 RETURN
550 IFILE=NTYP(I)
185 CALL NEXTWO(CARD,IST,NAM£JFLAG)
 IF(JFLAG) 520,560,540
560 CALL SWAPFIL(IFILE,NAM£)
 IF(IFILE.EQ.4.AND.5IFILE.EQ.5) GO TO 510
 IF(IFILE.EQ.4) GO TO 570
 REMIND 5
 IARCH=0
 GO TO 510
570 REMIND 4
 LASTP=0
195 JTOC=0
 GO TO 510
630 CONTINUE
 C REFUSE AND SELECT
 IF(JUMP.NE.19.AND.JUMP.NE.20) GO TO 700
200 NREC=0
 KFLAG=0
610 CALL NEXTWO(CARD,IST,NAM£JFLAG)
 IF(JFLAG.EQ.2) GO TO 640
 IF(JFLAG) 520,620,630
205 NREC=NREC+1
 IF(NREC.EQ.100) GO TO 650
 NAME=NREC,JNAME
 KFLAG=0
 GO TO 610
630 KFLAG=1
 GO TO 610
640 IF(KFLAG.EQ.0) RETURN
 READ 15,CARD
15 FORMAT(79A1)
215 PRINT 16,CARD
16 FORMAT(79A1)
 KFLAG=0
 IST=1
 GO TO 610
220 PRINT 17
SUBROUTINE ISIT

CDC 6630 FIM V3.0-P292 OPT=1 04

17 FORMAT(* EMPLIB $$$ MORE THAN 100 RECORD NAMES GIVEN -- TOO MANY."
JUMP=0
RETURN

225 CONTINUE

730 IF(JUMP.NE.22.AND.JUMP.NE.23) GO TO 930

C REWRITE AND ENDFILE

NAME=IB
CALL NEXTWCARD,IST,NAME,JFLAG)

230 IF(JFLAG.EQ.0) GO TO 710

PRINT 18,NAME,CARD

19 FORMAT(* EMPLIB $$$ ILLEGAL FILE TYPE.**,A10,***,79A1)

JUMP=0
RETURN

235 DO 720 I=1,9
IF(NAME.EQ.FILTP(I)) GO TO 730

720 CONTINUE

PRINT 14,NAME,FILTP
JUMP=0
RETURN

730 J=I
I=1
IF(JUMP.EQ.23) GO TO 750

IF(I.NE.6.AND.I.NE.7.AND.I.NE.4) GO TO 740

245 PRINT 19,CHAR(JUMP),FILTP(I)

19 FORMAT(* EMPLIB $$$ **,A10,** IS AN ILLEGAL DIRECTIVE FOR THE FILE T
"TYPE **,A2**,"

JUMP=0
RETURN

250 DIVISION I
NAME=FREWOUN-
D

245 PRINT 23,NAME,FILTP(I),FILNAM(I)

20 FORMAT(* EMPLIB $$$ *,A9,A2,** FILE NAMED *,A7,**)

RETURN

255 IF(I.NE.1.AND.I.NE.2) GO TO 735

ENDFILE I
NAME=ENDFILED
GO TO 745

900 CONTINUE

C SKIP AND SKPB

FILE=I
CALL NEXTWCARD,IST,NAME,JFLAG)

265 IF(JFLAG.EQ.0) GO TO 820

PRINT 21,NAME

21 FORMAT(* EMPLIB $$$ ILLEGAL NUMBR.**,A10)

JUMP=0
RETURN

270 FILE=I
I=SHIFT(MASK,54)

600 J=I
L=I.AND.NAME
IF(L.EQ.(I.AND.ID)) RETURN

275 DO 830 K=1,10

46
SUBROUTINE ISIT

IF (L.EQ.1.AND.SHIFT(VOLUME,66-5*J))) GO TO 840

830 CONTINUE
PRINT 21, NAME
JUMP=0
RETURN

840 IFILE=10*IFILE+K-1
850 I=SHIFT(I,64)
RETURN

900 CONTINUE

C REPLACE AND REPLACE
NAME0L=TOC(1,IFILE)
IVOLD=TOC(2,IFILE)
CALL NEXTWO(CARO,IST,NAME,JFLAG)
IF (JFLAG.NE.1) GO TO 215
NAME=IB
CALL NEXTWO(CARO,IST,NAME,JFLAG)
IF (JFLAG.EQ.3) GO TO 923, 249
IVERS=18
CALL NEXTWO(CARO,IST,IVERS,JFLAG)
IF (JFLAG.LT.0) GO TO 255
GO TO 260
END
SUBROUTINE NEXTWO

INTEGER CAR(I)

JFLAG=1 IS ERROR, 0 IS NORMAL RETURN, 1 IS COMMA, 2 IS EMPTY CARD
NAME=1H

JFLAG=2

IF(IST.GT.79) RETURN

IF(CARD(I).GE.1H) GO TO 40

IF(CARD(I).GE.1H) GO TO 15

CONTINUE

IST=80

RETURN

I=I+1

JFLAG=0

GO TO 11

IST=IST+11

IF(IST.GT.79) RETURN

IF(CARD(I).GE.1H) RETURN

CONTINUE

CALL APPEND(J,CARD(I),NAME)

JFLAG=-1

RETURN

IST=IST+1

JFLAG=1

RETURN

END
SUBROUTINE APPEND

SUBROUTINE APPEND(I, CHAR, X)
DATA MASK/7777/
ITEMP=SHIFT(MASK,60-6*I)
JITEMP=SHIFT(CHAR,6-6*I)
JITEMP=ITEMP .AND. JITEMP
X=X .AND. .NOT. JITEMP
X=X .OR. JITEMP
RETURN
END
SUBROUTINE CPYREC

SUBROUTINE CPYREC(IN,OUT,NAMREC,NREC)
COMMON /LMAX,A(1)
DIMENSION NAMREC(1)
INTEGER A
DATA N6K,776,I8/1H
COMMON N6K(NREC)
10 CONTINUE
BUFFER IN(IN1) (A,A(MAX))
10 IFUNIT(IN1) 30,20,10C
20 PRINT 1
1 FORMAT(50X,**END OF COPY**) RETURN
30 L=LENGTH(IN)
15 IF(L.EQ.0) GO TO 90
M=MAX(9X,L)
10 IF(L.LEC.0) Go TO 90
20 IF((A(2).AND.A(1).EQ.0) GO TO 50
IX=(IX.AND.90.63).OR.(A(1).AND.90)
40 CONTINUE
50 GO TO 80
25 IF(IX.EQ.NAMREC(1).AND.NREC.LT.0) GO TO 60
30 IF(IX.EQ.NAMREC(1).AND.NREC.GT.0) GO TO 70
25 IF(IX.EQ.NAMREC(1).AND.NREC.LT.0) GO TO 60
30 IF(IX.EQ.NAMREC(1).AND.NREC.GT.0) GO TO 70
60 BUFFER OUT(OUT1) (A,AIL)
30 IFUNIT(OUT1,CE.0) GO TO 110
PRINT 2,IX
2 FORMAT(45X,A7)
GO TO 10
70 PRINT 3,IX
3 FORMAT(60X,A7)
GO TO 10
80 CONTINUE
90 CONTINUE
40 PRINT 6
6 FORMAT(60X,**EMPTY RECORD ENCOUNTERED**)
GO TO 10
100 PRINT 4
4 FORMAT(* EMPLIE $$$ READ ERROR IN CPYREC.*)
CALL ABORT
110 PRINT 5
5 FORMAT(* EMPLIE $$$ WRITE ERROR IN CPYREC.*)
CALL ABORT
END