File No. S360-31
Form C28-6702-0

OS

IBM Systems Reference Library

IBM System/360 Operating System

Planning Guide for the Loader

This publication is a planning aid only. It is intended for use prior to the availability of the loader and shall be replaced by reference documentation when the loader becomes available.

The loader combines the basic editing and loading functions of the linkage editor and program fetch in one job step. It is designed for high performance loading of modules that do not require the special processing facilities of the linkage editor and fetch, such as overlay. The loader does not produce load modules for program libraries.

PREFACE

This publication is directed to the programmer who is planning to use the loader. The first part of the publication describes the functional characteristics of the loader; the second part describes the job control language statements and invocation procedures for the loader. All these items are discussed in relation to the facilities of the linkage editor. Therefore, the reader must be familiar with the publication IBM System/360 Operating System: Linkage Editor, Form C28-6538.

Also required for an understanding of this publication is IBM System/360 Operating System: Job Control Language, Form C28-6539.

First Edition (December 1968)

Requests for copies of IBM publications should be made to the IBM branch office serving your locality.

A form for readers' comments is provided at the back of this publication. If the form has been removed, comments may be addressed to IBM Corporation, Programming Systems Publications, Department D58, PO Box 390, Poughkeepsie, New York 12602

CONTENTS

ILLUSTRATIONS

FIGURES

TABLES

The Loader is one of the IBM System/360 Operating System processing programs. It combines basic editing and loading functions of the linkage editor and program fetch in one job step. Therefore, the load function is equivalent to the link edit-go function. The loader can be used for compile-load and load jobs. (If the linkage editor and fetch were used instead of the loader, a compile-load job would be called a compile-link edit-go job, and a load job would be called a link edit-go job.)

The loader will load object modules produced by a language processor and load modules produced by the linkage editor into main storage for execution. Optionally, it will search a call library (SYSLIB) or the link pack area queue of MVT, or both, to resolve external references. The loader does not produce load modules for program libraries.

The functional characteristics, compatibility and restrictions, performance considerations, and storage considerations of the loader are described in the following paragraphs.

FUNCTIONAL CHARACTERISTICS

The loader can be used with PCP, MFT, and MVT. The loader is reenterable and, therefore, it can reside in the link pack area of MVT.

The loader combines the following basic functions of the linkage editor and program fetch:

1. Resolution of external references between program modules.
2. Optional inclusion of modules from a call library (SYSLIB).
3. Automatic deletion of duplicate copies of program modules. (The first copy is loaded and all suceeding requests use that copy.)
4. Relocation of all address constants so that control may be passed directly to the assigned entry point in main storage. (Relocation of address constants is performed as the relocation dictionary (RLD) is encountered unless the referenced control section is not in main storage. In this case, the RLD is saved and relocated when the control section is loaded.)

The diagnostics produced by the loader are similar to those of the linkage editor.

COMPATIBILITY AND RESTRICTIONS

The loader accepts the same basic input as the linkage editor:

1. All object modules that can be processed by the linkage editor can be input to the loader.
2. All load modules produced by the linkage editor can be input to the loader (except load modules edited with the NE option).

The loader supports the following linkage editor options: MAP, LET, NCAL, AND SIZE. All other linkage editor options and attributes are not supported, but if used, they will not be considered as an error. A message will be listed on SYSPRINT indicating that they are not supported. The supported options are specified in the PARM field of the EXEC statement, or with the LINK, ATTACH, LOAD, or XCTL macro instruction. In addition to the supported linkage editor options, the loader provides several other options. All loader options are described in "EXEC statement" in the section "Planning for Use of the Loader."

The loader does not process linkage editor control statements (for example, INCLUDE, NAME, OVERLAY, etc.). If they are used, they will not be treated as an error and a message will be listed on SYSPRINT indicating that the control statements are not supported.

The loader and the linkage editor are bound by the same input conventions. (These conventions are discussed in the publication IBM System/360 Operating System: Linkage Editor.) In addition, the loader can accept load modules in the SYSLIN data set.

PERFORMANCE CONSIDERATIONS

The execution time of a problem program is the same whether it is processed by the loader or by the linkage editor and program fetch. However, the editing and loading time is greatly reduced when using the loader. Time savings are made by:

1. Reducing scheduling time.

2. Reducing equivalent linkage editor processing time.

3. Reducing I/O operations.

Scheduling time is reduced because the usual link edit-go steps are combined into one job step.

The loader can process a job in approximately half the time required by the linkage editor. This is done as follows:

1. Linkage editor intermediate and output I/O operations are eliminated.

2. Certain linkage editor functions, such as OVERLAY, are eliminated.

3. The I/O time required for reading load modules and object modules is reduced through the use of improved buffering techniques and through the use of chained scheduling, which reduces seek time between successive READs.

The processing time can be further reduced in a MVT environment by using link pack resident modules for the resolution of library references.

I/O operations and the amount of auxiliary storage required is reduced in compile-load and load jobs. For compile-load jobs, auxiliary storage space need not be allocated for linkage editor intermediate and output data sets. For production (load) jobs, the auxiliary storage space needed by the job library can be significantly reduced by deferring the inclusion of processor library routines in the program until load time. This deferment can also be made through the NCAL option of the linkage editor, but the equivalent processing is faster using the loader, especially if the loader uses the link pack area of MVT to resolve external references.

STORAGE CONSIDERATIONS

The loader requires main storage space for the following items:

- Loader code.
- Data management access methods.
- Buffers and tables used by the loader (dynamic storage).
- Loaded program (dynamic storage).

Table 1 shows the appropriate storage requirements in bytes.

All or part of the main storage required is obtained from user storage. If the access methods are made resident at IPL time, they are allocated in system storage. In an MVT environment the loader code could also be made resident in the link pack area.

The sizes of the buffers and tables are dynamically expanded to use the available main storage specified with the SIZE parameter.

The size of the loaded program is the same as if the program had been processed by the linkage editor and program fetch.

The loader does not use auxiliary storage space for work areas.

Table 1. Main Storage Requirements

Consideration	Approximate Main Storage Requirement in Bytes	Comments
Loader Code	10K	
Data Management	4K	BSAM
Fixed SYSLIN Buffer and DECBs	BUFNO(BLKSIZE + 24)	Concatenation of different BLKSIZE and BUFNO must be considered. (Minimum BUFNO=2)
Load Module Buffer Size & DECBs	304	
SYSPRINT Buffer and DECBs	BUFNO(BLKSIZE + 24)	Buffer size rounded up to integral number of double words. (Minimum BUFNO=2)
Size of program being loaded	Program Size	Program size is restricted only by available main storage
Each external relocation dictionary entry	8	
Each external symbol	20	
Largest ESD number	4n n is the size of the largest ESD in any module	Allocated in increments of 32 entries
Fixed Loader Table Size	1202	Subtract 88 if NOPRINT is specified

PLANNING FOR USE OF THE LOADER

This section discusses how to prepare an input deck for the loader and how to invoke the loader.

INPUT FOR THE LOADER

The input deck for the loader must contain job control language statements for the loader and, optionally, for the loaded program (see Figure 1).

```
//name      JOB  parameters           (optional)
//name     EXEC  PGM=LOADER,PARM=(parameters)
//SYSLIN    DD   parameters
//SYSLIB    DD   parameters           (optional)
//SYSPRINT  DD   parameters           (optional)
//     (optional DD statements and data required for loaded program)
```

Figure 1. Input Deck for the Loader

Only the EXEC statement and the SYSLIN DD statement are required for a loader step. The JOB statement is required if the loader is the first step in the job.

EXEC STATEMENT

The EXEC statement is used to call the loader and to specify options for the loader and for the loaded program. The loader is called by specifying PGM=IEWLDRGO or PGM=LOADER (see "Invoking the Loader"). Loader and loaded program options are specified in the PARM field of the EXEC statement. The PARM field must have the following format:

```
,PARM='[loaderoption[,loaderoption]...][/programoption
                                        [,programoption]...]'
```

Note that the loaded program options, if any, must be separated from the loader options by a slash (/). If there are no loader options, the program options must begin with a slash. The entire PARM field may be omitted if there are no loader or loaded program options.

All parameters must be enclosed in apostrophes when special characters (/ and =) are used.

The loader options are:

MAP
    The loader produces a map of the loaded program that lists external names and their absolute storage addresses on the SYSPRINT data set. (If the SYSPRINT DD statement is not used in the input deck, this option is ignored.) An example of a module map is shown in Figure 2.

NOMAP
    A map is not produced.

RES

An automatic search of the link pack area queue is to be made. This search is always made after processing the primary input (SYSLIN), and before searching the SYSLIB data set.

NORES

An automatic search of the link pack area queue is not to be made.

CALL

An automatic search of the SYSLIB data set is to be made. (If the SYSLIB DD statement is not used in the input deck, this option is ignored.)

NOCALL
or NCAL

An automatic search of the SYSLIB data set will not be made.

LET

The loader will try to execute the object program even though a severity 2 error condition is found. (A severity 2 error condition is one that could make execution of the loaded program impossible.)

NOLET

The loader will not try to execute the loaded program when a severity 2 error condition is found.

SIZE=size

specifies the size, in bytes, of dynamic main storage that can be used by the loader. Normally, this value will be 17K plus the size of the program to be loaded. (See "Storage Considerations".)

EP=name

specifies the external name to be assigned as the entry point of the loaded program.

PRINT

Diagnostic messages are produced on the SYSPRINT data set.

NOPRINT

Diagnostic messages will not be produced on the SYSPRINT data set. SYSPRINT will not be opened

Unless otherwise specified with the LOADER macro instruction during system generation, the default options are: NOMAP, CALL, NOLET, SIZE=100K, PRINT, and RES.

The following are examples of the EXEC statement. In these examples, X and Y are parameters required by the loaded program.

```
//LOAD    EXEC    PGM=LOADER
//LOAD    EXEC    PGM=IEWLDRGO,PARM='MAP,EP=FIRST/X,Y'
//LOAD    EXEC    PGM=LOADER,PARM='/X,Y'
//LOAD    EXEC    PGM=IEWLDRGO,PARM=(MAP,LET)
//LOAD    EXEC    PGM=LOADER,PARM=NOPRINT
```

For further details in coding the EXEC statement refer to the publication IBM System/360 Operating System: Job Control Language.

Figure 2. Module Map Format Example

```
                                                  OS/360 LOADER

OPTIONS USED - PRINT,MAP,NOLET,CALL,NORES,SIZE=424176

     NAME   TYPE   ADDR         NAME   TYPE   ADDR         NAME   TYPE   ADDR         NAME   TYPE   ADDR         NAME   TYPE   ADDR

   SAMPL2B    SD    161E0     SAMPL2BA   SD    16EC8     IHEMAIN    SD    17CF8     IHENTRY    SD    17D00     IHESPRT    SD    17D10
   SYSIN      SD    17D48     IHEVQC   * SD    17D80     IHEVQCA  * LR    17D80     IHEVQB   * SD    17FD8     IHEVQBA  * LR    17FD8
   IHEDIA   * SD    183C0     IHEDIAA  * LR    183C0     IHEDIAB  * LR    183C2     IHEVPE   * SD    18608     IHEVPEA  * LR    18608
   IHEVPA   * SD    18870     IHEVPAA  * LR    18870     IHEVFC   * SD    189D0     IHEVFCA  * LR    189D0     IHEVPC   * SD    189F8
   IHEVPCA  * LR    189F8     IHEVFE   * SD    18BE8     IHEVFEA  * LR    18BE8     IHEVSC   * SD    18C08     IHEVSCA  * LR    18C08
   IHEDNC   * SD    18CB8     IHEDNCA  * LR    18CB8     IHEDOA   * SD    18F30     IHEDOAA  * LR    18F30     IHEDOAB  * LR    18F32
   IHEDMA   * SD    19010     IHEDMAA  * LR    19010     IHEVFD   * SD    19108     IHEVFDA  * LR    19108     IHEVFA   * SD    19160
   IHEVFAA  * LR    19160     IHEVPB   * SD    19248     IHEVPBA  * LR    19248     IHEXIS   * SD    193F0     IHEXISO  * LR    193F0
   IHEIOB   * SD    19488     IHEIOBA  * LR    19488     IHEIOBB  * LR    19490     IHEIOBC  * LR    19498     IHEIOBD  * LR    194A0
   IHESARC  * LR    1A9C8     IHESADD  * LR    1A9DE     IHESAFF  * LR    1AA18     IHEPRT   * SD    1AB70     IHEPRTA  * LR    1AB70
   IHEBEGA  * LR    1AE28     IHEERR   * SD    1AE68     IHEERRD  * LR    1AE68     IHEERRC  * LR    1AE72     IHEERRB  * LR    1AE7C
   IHEERRA  * LR    1AE86     IHEERRE  * LR    1B4E2     IHEIOF   * SD    1B580     IHEIOFB  * LR    1B580     IHEIOFA  * LR    1B582
   IHEITAZ  * LR    1B81E     IHEITAX  * LR    1B82A     IHEITAA  * LR    1B83E     IHEDCN   * SD    1B860     IHEDCNA  * LR    1B860
   IHEDCNB  * LR    1B862     IHEIOD   * SD    1BA50     IHEIODG  * LR    1BA50     IHEIODP  * LR    1BA52     IHEIODT  * LR    1BB4A
   IHEVTB   * SD    1BCF0     IHEVTBA  * LR    1BCF0     IHEVQA   * SD    1BD78     IHEVQAA  * LR    1BD78



   IHEQINV    PR       00     IHEQERR    PR        4     SAMPL2BB   PR        8     SAMPL2BC   PR        C     IHEQSPR     PR       10
   SYSIN      PR       14     IHEQLSA    PR       18     IHEQLW0    PR       1C     IHEQLW1    PR       20     IHEQLW2     PR       24
   IHEQLW3    PR       28     IHEQLW4    PR       2C     IHEQLWE    PR       30     IHEQLCA    PR       34     IHEQVDA     PR       38
   IHEQFVD    PR       3C     IHEQCFL    PR       40     IHEQFOP    PR       48     IHEQADC    PR       4C     IHEQXLV     PR       50
   IHEQEVT    PR       58     IHEQSLA    PR       60     IHEQSAR    PR       64     IHEQLWF    PR       68     IHEQRTC     PR       6C
   IHEQSFC    PR       70

IEW1001  IHEUPBA
IEW1001  IHEUPAA
IEW1001  IHETERA
IEW1001  IHEM91C
IEW1001  IHEM91B
IEW1001  IHEM91A
IEW1001  IHEDDOD
IEW1001  IHEVPFA
IEW1001  IHEVPDA
IEW1001  IHEDBNA
IEW1001  IHEVSFA
IEW1001  IHEVSBA
IEW1001  IHEVCAA
IEW1001  IHEVSAA
IEW1001  IHEDNBA
IEW1001  IHEUPBB
IEW1001  IHEUPAB
IEW1001  IHEVSEB

  TOTAL LENGTH      5068
  ENTRY ADDRESS     17D00
```

DD STATEMENTS

The loader uses three DD statements named SYSLIN, SYSLIB, and SYSPRINT. (These ddnames can be changed during system generation with the LOADER macro instruction.) The SYSLIN DD statement must be used in every loader job. The other two are optional. The data sets defined by the SYSLIN DD statements can be either sequential data sets or members of a partitioned data set. The data set defined by the SYSLIB DD statement must be a partitioned data set. The data set defined by the SYSPRINT DD statement must be a sequential data set. Concatenation may be used to include multiple input data sets in SYSLIN and SYSLIB. For better performance, the user can specify the number of buffers and the block record size for SYSLIN, SYSLIB, and SYSPRINT. In any case, a minimum of two buffers are allocated to each data set. Chained scheduling can be used for data sets with fixed record format.

The SYSLIN data set contains the input to the loader. This input can be either object modules produced by a language translator, or load modules produced by the linkage editor, or both. If concatenated data sets are used, the DD statement for the data set with the largest blocksize should appear first in the sequence for more efficient utilization of main storage.

The SYSLIB data set contains IBM or user-written library routines to be included in the loaded program. The SYSLIB data set is searched when unresolved references remain after processing SYSLIN and optionally searching the link pack area. The library may contain either object modules or load modules, but not both.

The SYSPRINT data set is used for error and warning messages and for an optional map of external references. The record format of SYSPRINT must be FA, FBA, or FBSA.

In addition to the DD statements used by the loader, any DD statements and data required by the loaded program must be included in the input deck.

LOADED PROGRAM DATA

Loaded program data and loader data can both be specified in the input reader. In MFT and MVT, loaded program data can be defined by a DD statement following the loader data. In PCP, both loader data and loaded program data must be defined by the same DD statement. As shown in Figure 3, this is accomplished by using the AFF subparameter of the UNIT parameter in the DD statement that defines input data for the loaded program.

```
//LOAD       JOB   MSGLEVEL=1
//LDR       EXEC   PGM=LOADER,PARM=MAP
//SYSLIB     DD    DSNAME=SYS1.FORTLIB,DISP=SHR
//SYSPRINT   DD    SYSOUT=A
//SYSLIN     DD    DDNAME=SYSIN
//FT06F001   DD    SYSOUT=A
//FT05F001   DD    UNIT=AFF=SYSLIN,DCB=(LRECL=80,RECFM=F)
//SYSIN      DD    *
                 .)
                 .}Loader input data
                 .)
/*
                 .)
                 .}Loaded program data
                 .)
/*
```

Figure 3. Loader and Loaded Program Data in PCP Input Stream

SAMPLE INPUT FOR THE LOADER

Figure 4 shows an input deck for a load job. A previously assembled program, MAIN, is to be loaded. The SYSPRINT and SYSLIB DD statements are not used.

```
//LOAD       JOB   MSGLEVEL=1
//        EXEC   PGM=LOADER
//SYSLIN     DD    DSNAME=MAIN,DISP=OLD

   DD statements and data required for execution of MAIN

/*
```

Figure 4. Input Deck for a Load Job

Figure 5 shows an input deck for a compile-load job. The FORTRAN E (IEJFAAA0) compiler is used for the compile step.

```
//JOB         JOB   22,MCS,MSGLEVEL=1
//FORT       EXEC   PGM=IEJFAAA0,PARM='MAP,LIST',REGION=52K,RD=R
//SYSPRINT    DD    SYSOUT=A
//SYSPUNCH    DD    UNIT=SYSCP
//SYSUT1      DD    UNIT=SYSSQ,SPACE=(TRK,(30,10))
//SYSUT2      DD    UNIT=SYSSQ,SPACE=(TRK,(30,10))
//SYSLIN      DD    DSNAME=&LOADSET,DISP=(MOD,PASS),                   X
//                  UNIT=SYSSQ,SPACE=(TRK,(30,10))
//SYSIN       DD    *
                .)
                .}Source program
                .)
/*
//LOAD       EXEC   PGM=LOADER,PARM='MAP,LET',COND=(5,LT,FORT)
//SYSLIN      DD    DSNAME=*.FORT.SYSLIN,DISP=(OLD,DELETE)
//SYSPRINT    DD    SYSOUT=A
//SYSLIB      DD    DSNAME=SYS1.FORTLIB,DISP=SHR
//FT02F001    DD    SYSOUT=B
//FT03F001    DD    SYSOUT=A
//FT01F001    DD    DDNAME=SYSIN
//SYSIN       DD    *
                .)
                .}Data for Loaded Program
                .)
/*
```

Figure 5. Input Deck for a Compile-Load Job

INVOKING THE LOADER

The loader can be referred to by either its program name, IEWLDRGO, or its alias, LOADER. The loader can be invoked through the EXEC statement, as described in "Input for the Loader," or through the LOAD, ATTACH, LINK, or XCTL macro instruction. Figure 6 shows the basic format for the macro instruction.

Name	Operation	Operand

[symbol]	{LINK
	ATTACH}
		EP=loadername
		PARAM=(optionlist[,ddname list])
		VL=1

	{LOAD XCTL}	EP=loadername

Figure 6. Macro Instruction Basic Format

If desired, the user may invoke the loader to process a program but not execute it. This can be accomplished by referring to the name IEWLOADR with a LOAD and CALL macro instruction. (IEWLOADR is the portion of the loader that processes the loader input data.) The entry point of the loaded program is returned in register 0. Register 1 points to two full words: the first points to the beginning of storage occupied by the loaded program; the second contains the size of the loaded program. This location and size can then be used in a FREEMAIN macro instruction to free the storage occupied by the loaded program when it is no longer needed.

For further information on the use of these macro instructions, refer to the publication IBM System/360 Operating System: Supervisor and Data Management Services, Form C28-6646.

IBM®

International Business Machines Corporation
Data Processing Division
112 East Post Road, White Plains, N.Y. 10601
[USA Only]

IBM World Trade Corporation
821 United Nations Plaza, New York, New York 10017
[International]

READER'S COMMENT FORM

IBM System/360 Operating System
Planning Guide for the Loader

Form C28-6702-0

- Is the material:

	Yes	No
Easy to read?	□	□
Well organized?	□	□
Complete?	□	□
Well illustrated?	□	□
Accurate?	□	□
Suitable for its intended audience?	□	□

- How did you use this publication?

□ As an introduction to the subject
□ For additional knowledge
Other ....

- Please check the items that describe your position:

□ Customer personnel
□ IBM personnel
□ Manager
□ Systems Analyst
□ Operator
□ Programmer
□ Customer Engineer
□ Instructor
□ Sales Representative
□ Systems Engineer
□ Trainee
Other ....

- Please check specific criticism(s), give page number(s), and explain below:

□ Clarification on page(s) ....
□ Addition on page(s) ....
□ Deletion on page(s) ....
□ Error on page(s) ....

Explanation:

- Thank you for your cooperation. No postage necessary if mailed in the U.S.A.

YOUR COMMENTS PLEASE . . .

This publication is one of a series which serves as reference for systems analysts, programmers and operators of IBM systems. Your answers to the questions on the back of this form, together with your comments, will help us produce better publications for your use. Each reply will be carefully reviewed by the persons responsible for writing and publishing this material. All comments and suggestions become the property of IBM.

Please note: Requests for copies of publications and for assistance in utilizing your IBM system should be directed to your IBM representative or to the IBM sales office serving your locality.

Fold Fold

FIRST CLASS
PERMIT NO. 81
POUGHKEEPSIE, N.Y.

BUSINESS REPLY MAIL
NO POSTAGE STAMP NECESSARY IF MAILED IN U. S. A.

POSTAGE WILL BE PAID BY

IBM Corporation
P.O. Box 390
Poughkeepsie, N.Y. 12602

Attention: Programming Systems Publications
Department D58

Fold Fold

IBM®

International Business Machines Corporation
Data Processing Division
112 East Post Road, White Plains, N.Y. 10601
[USA Only]

IBM World Trade Corporation
821 United Nations Plaza, New York, New York 10017
[International]