1120

Bell Laboratories

subject: DOS-BATCH to UNIX Conversions - Case 40979

date: September 1, 1976

from: E. H. Albrecht
5161-760901.01MF

MEMORANDUM FOR FILE

These notes and the associated attachment were written to arm the knowledgeable DOS-BATCH programmer with sufficient background to permit him to develop programs, executable on a "stand alone" PDP 11 processor, using the UNIX operating system. The DOS-BATCH system of Department 5161 is currently used in the mode where the machine is dedicated to one individual at a time. The UNIX environment supports up to 14 simultaneous users. Because UNIX is a time sharing system, it may also be accessed via remote terminals. Therefore, a program may be edited, assembled and linked remotely, providing a suitable terminal is available. Load modules may be transmitted to "stand alone" processors for execution. These arrangements allow for extensive program modifications to be made and tested from the work location, at home or field trial sites. The overall affect of such arrangements, for Department 5161, should be to increase programming flexibility and programmer productivity measurable through an increase in computing resources.

The following highlights various aspects of a DOS to UNIX conversion procedure, and documents my encounters in the conversion process. It is intended to be used more as a directory rather than a tutorial.

DOCUMENTATION

The UNIX operating system is documented in the "UNIX Programmer's Manual" which is divided, via separaters, into many parts. The CONTENTS section of the UNIX Manual briefly describes the commands, functions, etc., contained in the manual. The REFERENCE section of the manual contains several memos, articles, etc., that describe UNIX in greater detail.

The majority of utilities (commands) the beginning programmer will use are located in Section (I) or 1. A command shown in the notation "MKDIR(I)" would indicate that it is to be found in Section 1 or I. Within Section (I), commands are ordered alphabetically.

The DOS-BATCH simulation program documentation for our system (Department 5161) is located in a UNIX directory /USR/DOS. The documentation for specific programs is located in files within this directory. This information is included in the Attachment DOCUMENT.

GENERAL CONSIDERATIONS

Control Signals

DOS-BATCH memory management commands, "Control C", ".KI", "Control U", ("Rubout") and/or "Delete" are not needed in UNIX and are accomplished via other means and have other meanings in a UNIX environment. The following list summarizes the DOS-BATCH control signals and indicates the equivalents.

- "@"<CR> = "Control U"

- "#" = "Delete"

- "Rub out" or "Delete" means halt the current UNIX operation or UNIX function that is active and to return control to the user.

- "Control D" is used to abort a UNIX editing function or to end a terminal session (i.e., logging out).

- The "ALT" or "ESC" key may be used to momentarily halt printing (or video display) at a UNIX terminal.

- The "#" or "$" that DOS-BATCH operation systems use to indicate a ready state are replaced by the UNIX "%" symbol.

- The "*" symbol that the DOS-BATCH editor uses to indicate a ready condition is not replaced unless the DEC VT05 video terminal is used.

When editing, via UNIX, you must precede the use of "#", "$", "@", etc., with a slash "/" when these symbols are to appear as text.

A loose equivalence between the DOS-BATCH operating system "UICs" and UNIX "directories" may be made. A directory may be created via the "MKDIR(I)"* command and removed via the "RMDIR(I)"* command. Transferring operations from one UNIX directory to another requires a "CHDIR(I)" command and does not require the DOS-BATCH sequence of "Control C", ".KI" and "FI" commands followed by the new log-in sequence.

* See UNIX Programmers Manual Section 1.

UNIX operating system file and directory interactions are based on a tree structure. All directories of the UNIX operating system are considered higher level nodes in the tree structure (e.g., they may have subnodes appended to them), while all data files are considered as end nodes in the tree structure (e.g., they cannot have any subnodes appended to them). Changing operations from one directory to another or moving files from one directory to another requires the user to define the path through the tree structure, from the highest node to the least node, that will be traversed. This path must start in the tree structure at a node common to both directories (Root node or the current directory being used). The path descends through the tree structure until the desired node (directory) is reached. In all cases, the ROOT ("/") or trunk node is common to all nodes.

The following example will show the use of path name notation to show how a user may shift from one directory to another.

> A slash "/" assumes that a directory name or a file name follows. The first "/" in a character string defines the trunk node or "ROOT" of the tree.
> Example: Assume two subnodes, "source" and "object" are appended to a user directory node "erwin".
> Assume that the user logs in under the name "erwin".

Example: `%MKDIR SOURCE <cr>;`

> Create a directory called "source". This new directory will be a node appended to directory "erwin".

Example: `%CHDIR SOURCE <cr>`

> Change operations to directory "source". The user wants to operate on a file in the "source" directory. He uses this command to switch operations from the directory "erwin" (he logged in under directory "erwin") to the directory "source" which is a subnode of the "erwin" directory. (See UNIX Manual CHDIR (I)).

Example: `%CHDIR /USR/ERWIN/ØBJECT <CR>`

> The user switches operations from the directory "source" (from the previous operation) to the top of the tree or root node which is common to the tree structure and is directed down the tree structure to the desired directory node "object". The "object"

directory would have to have been created previously via the MKDIR(I)* command. The full path name was used because it is unambigous.

Example: %MV FILEA /USER/ERWIN/SOURCE

The user is in any directory, possible even another user's directory or subdirectory, and transfers a file from this current working directory to the directory "source". (See UNIX Manual MV(I).)

A terminal which connects to a UNIX operating system uses the lower case ASCII as normal input mode. A UNIX utility command "DD(I)"* provides easy conversion of source files between the upper case ASCII used in the DOS system and lower case ASCII so the editing via UNIX is not as troublesome. To convert a source file (MY.SRC) to lower case:

```
%DD IF=MY.SRC OF=TMP   CONV=LCASE

%MV TMP MY.SRC
```

In this example, TMP was used as an intermediate file to hold the lower case equivalent of MY.SRC. The MV utility transferred TMP back to the MY.SRC file.

To convert a source file the (MY.SRC) to upper case:

```
%DD IF=MY.SRC  OF=TMP     CONV=UCASE

%MV TMP  MY.SRC
```

The general form of the DOS-BATCH command shows the "TO" file first and follows with the "FROM" file. The DOS-BATCH commands are also a subset of a general utility program such as PIP, MACRO, etc. Under UNIX, the command contains a mnemonic for the utlity plus the "TO" and "FROM" file specifications. The UNIX command lists the "FROM" file first and it is followed by the "TO" file. The commands required in both systems to transfer a file are shown in the following example:

Example:	DOS-BATCH	UNIX (EQUIVALENT)
	"Control C"	%MV FROM.SRC TO.SRC
	.KI	or
	$RU PIP	%CP FROM.SRC TO.SRC
	#TO.SRC<FROM.SRC	

*See UNIX Programmers Manual.

(See UNIX Manual CP(I) or MV(I)).

DOS-BATCH commands and the majority of DOS-BATCH work is done in upper case text. UNIX will accept upper case text if your login response was in upper case. UNIX will then convert all text, response from commands or contents of files etc, to upper case temporarily until you log off. UNIX will accept commands in lower case if your login response is in lower case.

UNIX LOGIN PROCEDURE:

Step 1) Arrange to have UNIX prepared to accept your logon or use an existing identifier. The person in charge of the UNIX should take care of this.

Step 2) Turn on a terminal attached to the UNIX System.

Step 3) Operate the carriage return <cr> or alternatively, operate the "CTRL" key simultaneously with the letter 'd'.

Step 4) When UNIX responds with 'LOGIN', you type your identifier.

Step 5) UNIX will respond with the message of the day.

Step 6) UNIX will print a '%' percent sign on your terminal to indicate that it is ready to respond to your commands.

See attachment for a login example.

PIP

Special programs ('PIPTP' and 'PIPRK') are used to move files between DOS-BATCH and UNIX. They are described in the attachment DOCUMENT. The following is to be used as a ready conversion table to permit an easy cross reference between common PIP functions and their equivalent UNIX functions. The list is not exhaustive and only one of a variety of equivalences is shown.

The control sequence of 'Control C', '.KI', and 'RU PIP' is not a required prelude to UNIX commands.

DOS-BATCH	UNIX
#/DI	%LS -L
#/BR	%LS
#LP:</DI	%LS -L ^ LPR
#BLAH.SRC/DE	%RM BLAH.SRC
#BLAH.SRC<BUNK.SRC	%CP BUNK.SRC BLAH.SRC
#LP:<BLAH.SRC	%LPR BLAH.SRC
#*.SRC/DE	%RM *.SRC
#DTO:/ZE	%TP OC
#DTO:<BLAH.SRC	%TP RO BLAH.SRC
#MTO:<BLAH.SRC	%TP ROM BLAH.SRC
#BLAH.SRC<DTO:BLAH.SRC	%TP XO BLAH.SRC
#LP:<DTO:/DI	%TP OT ^ LPR

See the UNIX Manual for LS(I), RM(I), CP(I), LPR(I), and TP(I).

EDIT

The UNIX editor is used to edit files. The terminals you use to interface with UNIX operate primarily in lower case ASCII. Your source code will normally be in upper case ASCII. For editing, it would be wise to use the UNIX 'DD(I)' command to convert a file to lower case prior to editing to lessen the frustration of locating character strings. See UNIX Manual for DD(I) and ED(I) to get more details. The following example provides a rough equivalence between the DOS-BATCH editor and the UNIX editor.

```
DOS-BATCH                     UNIX

#BLAH.SRC<BLAH.SRC            %ED BLAH.SRC
*R
*B 2L                         _1,2P
*I                            _I
    TEXT                           TEXT
   ‡ TEXT                            TEXT
<LF>                          <.>‡
*G/TEXT/V                     _/TEXT/P
*G/TEXT/ -4J 4C/WORD/V        _/TEXT/S/TEXT/WORD/P

*-A 2K                        {_.-1
                              {_.,.+1D

*EX                           {_W
                              {_Q
```

MACRO

The UNIX macro assembler takes a source file as input and produces two output files; an object file (.OBJ) and a

‡ <LF> means operate DATE FILE COPY key. For UNIX The period "." is the equivalent of <LF>.

listing file (.LST). To get assembly listings requires that you direct the .LST file to a printer. See Attachment DOCUMENT for MACRO documentation. Macro:

a) requires that all .GLOBLs to be explicitly declared

b) takes upper or lower case ASCII input.

The following table lists the UNIX macro commands and options along with their DOS-BATCH equivalents.

DOS-BATCH	UNIX
#BLAH.OBJ<BLAH.SRC	%MACRO BLAH.SRC
#BLAH.OBJ,LP:<BLAH.SRC	%MACRO -LS BLAH.SRC %LPR BLAH.LST } one of these %CAT BLAH.LST }
%BLAH.OBJ,LP:/CRF<BLAH.SRC	%MACRO -LS -CR BLAH.SRC %LPR BLAH.LST

LINKER

The UNIX link simulator, LINKER, takes one or more object files as input and produces two output files. The output files have the same high level qualifier name as the last object module input file. The output files are '.MAP' and '.OUT'. The MAP file is only produced if the '-LS' option is invoked. The load module file, .OUT is not executable under either DOS-BATCH or UNIX. It must be converted via the %LMC* command prior to being used on a separate processor. The following table should be helpful to show the UNIX equivalents to DOS-BATCH commands.

DOS-BATCH	UNIX
#BLAH.LDA<A.OBJ,B.OBJ/E	%LINKER A.OBJ B.OBJ %LMC B.OUT B.LDA
#BLAH.LDA,LP:<A,B,C/E	%LINKER A B C -LS %LMC C.OUT BLAH.LDA %LPR C.MAP
#BLAH.LDA/CRF,LP:<A,B,F/B:1000/E	%LINKER A B F -LS -CR -B:1000 %LMC F.OUT BLAH.LDA %LPR F.MAP

* See Attachment DOCUMENT for LINKR and LMC documentation.

UNIX currently does not have a DOS-BATCH library simulator. The library routine is used in DOS-BATCH primarily to eliminate typing several lines of text when invoking the DOS-BATCH link program. The same result (e.g., elimination of excessive lines of typing) may be accomplished under UNIX via a shell* procedure. Another alternative is to use the indirect list option when invoking the LINKR. An indirect list consists of a UNIX file that contains a list of file names. When LINKR is invoked, the files listed in the indirect list file are added to the load module being formed. LINKR has a bottom switch but has no top switch. If any absolute address commands are specified to the assembler (e.g., . = .600 or .ASECT) and these addresses lie below the bottom switch value specified, no load module can be formed. Alternatively, an ASECT module is inserted as the first module to be linked. The size of the .ASECT is equal to the bottom switch value that normally would be specified.

ABOUT SHELLS

A shell is nothing more than a file. The file contains UNIX commands. The file, when executed, sends the commands to UNIX one at a time. UNIX then executes these commands. The file containing the UNIX commands is called a 'shell'. On the IBM 370 it is called a catalogued procedure and/or a command procedure.

A shell may be created via the UNIX editor, it is stored in your own directory. The shell may be executed via the SH(I) command as follows:

```
%sh filename
```

If I let my imagination soar, I could envision a project containing some dozen files. I could with one shell, MACRO assemble all the files and another shell would be used to link all the resultant object modules. The power of the shell almost eliminates the need of a library and associated linking library search when many object modules are involved. Clearly, a shell is powerful tool.

LOAD MODULES

The output from the LINKR 'out' file, is not executable and must be converted to a '.LDA' file via the LMC(I)† command.

```
    %LMC                BLAH.OUT              BLAH.LDA
```

* See SH(I) in UNIX manual - Also, see next section.

† See Attachment

The '.LDA' file cannot be executed in a UNIX machine without conflicting with the UNIX operating system. The '.LDA' load module is, in general, transmitted to a 'stand alone' processor for execution. For Department 5161, we transmit the load module via a telephone line or dedicated data link by using the RLOAD* command. Program execution or program debugging is then accomplished on the 'stand alone' processor.

The 'stand alone' processor needs to be conditioned prior to receiving a load module. A separate program is available† for receiving a load module for:

a) A DL11 data link interface at the receiving computer.

b) a DC11 data link interface at the receiving computer for:

   1) 300 band data transmission speed. (telephone line)

   2) 1800 band data transmission speed (dedicated data link).

HOW TO PUT IT ALL TOGETHER

1) Use DOS-BATCH PIP to put your source files into a magtape or dec-tape. (I recommend you leave your object modules and load modules behind.) Optionally, you may use PIPRK to get files directly from disk and jump to Step 3.

2) Unload the source file tape into UNIX via the PIPTP command. (See Attachment DOCUMENT for PIPTP.)

3) Assemble the source files. (See MACRO in Attachment DOCUMENT.)

4) Link the object modules. (You do not need a shell until you get tired of typing.) (See Attachment DOCUMENT.)

5) Convert the '.OUT' file to a '.LDA' file via LMC. (See Attachment DOCUMENT for LMC.)

6) Condition the debugging computer to receive the load module, (e.g., load and execute the RLOAD.LDA cassette in the receiving PDP 11 processor (PDP 11/20 or PDP 11/10 or a UNIPRO).

____

* See Attachment DOCUMENT

† See Lee Moffitt, Ed Lyons or Erwin Albrecht for a copy of this program.

7) Set up a data link. Direct data links are always ready. (If a phone connection is needed, this is a separate step.) (If a direct link already exists, this step may be excluded.)

8) Transmit the load module to the other computer via: %RLOAD BLAH.LDA XX YY (See Attachment DOCUMENT or DOS(I).)

9) Execute or debug the program.

Credits

The conversion procedure described were only possible through the efforts of a lot of other people. Primarily, Fred Six who was always ready to lend a hand and gave advice on the overall project. To John Barberio for support on I/O routines. To Paul Jensen for support on the DOS-BATCH simulation programs. To J. Maranzano, et al for DOS-BATCH to UNIX I/O device and software support. To G. Fuchs for hardware support in the lab. To Meg Crowe for software librarian services support.

SUMMARY

A process whereby DOS programs may be developed under a UNIX operating system is described. The descriptions builds on the knowledge held by a DOS programmer and provides him with sufficient information to effect a programmer development transformation from a DOS operating system environment to a UNIX operating system environment. The UNIX operating system provides a flexibility that permits program development to be effected from the system console, from a terminal at a field trial site or at home.

E. H. Albrecht

HO-5161-EHA-vw E. H. Albrecht

Att.
Attachment DOCUMENT

Copy to
L. Emerson)
R. W. Ginn) WE
G. Hogfelt)
W. R. Lawler)

All Supervision 516
All Members Switching Service Improvement Group

Copy to (continued)
See next page

Copy to (continued)
J. Barberio
M. M. Crowe
D. M. Feste
G. E. Fuchs
P. Jensen
J. Marizano
F. B. Six

Attachment DOCUMENT

This attachment contains partial documentation for DOS simulations routines and utilities.

```
%

%
```

Prompts from previous terminal session

```
%
@LOGIN: erwin
```

USER TYPED HIS IDENTITY.

USER TYPED A 'CTRL' & a 'd'

```
UNIX 11/45 *** 3.0+  (I&D)
UNIX/DOS Disk: 12-04-75
SCCS, multi-file TM driver
Macro-compatible C compiler and library
%

%
```

UNIX MESSAGE ØF THE DAY

UNIX RESPONDS WITH a 'READY' PROMPT.

SYNOPSIS
A UTILITY TO MOVE A DOS FILE TO A UNIX DIRECTORY.

DESCRIPTION
A DOS FORMATTED FILE LOCATED ON A DOS FORMATTED DISK CARTRIDGE IS MOVED TO A UNIX DIRECTORY AND CONVERTED INTO A UNIX FORMATTED FILE.
THE FILE IS MOVED INTO THE UNIX DIRECTORY FROM WHICH THE 'PIPRK' COMMAND WAS ISSUED.

DIAGNOSTICS
ERROR AND INFORMATIVE MESSAGES ARE PROVIDED AS NEEDED.

BUGS
PIPRK WILL NOT MOVE A UNIX FILE UNTO A DOS DISK.

EXAMPLE

```
% piprk *

Are you going 'to DOS' ? no *

UNIX file name = my.src *

DOS file name = you.src *

DOS UIC = 4 15 *

Linked or Contiguous DOS file ? linked *

DOS structured disk in rk1 ? yes *

DOS disk statistics ? no *
```

* ~~underlined~~
user responses are underlined.

NAME

piptp - manipulate PIP format magtapes under UNIX

SYNOPSIS

piptp -rlw[n] [name ...]

DESCRIPTION

Piptp enables the user to read magnetic tape files which were generated under DOS in PIP format. These files are stored as UNIX files for editing under UNIX and then may be rewritten on magnetic tape in PIP format. Files which have been created under UNIX may also be written out in PIP format. The list option may be used in conjunction with the read or write options.

-r reads files from a PIP format magnetic tape into the current directory. The first record of each file is a 7 word header which includes the file name and extension stored in RADIX50 format. The file name and extension are extracted and the rest of the header information is discarded. If that file does not exist, it is created. If it already exists, it is overwritten. If no file names are input, the entire tape is read.

-l lists the file names from a PIP format tape.

-w writes UNIX files onto magnetic tape in PIP format. File names may be specified in any way acceptable to UNIX including the use of path names. File names are restricted to six characters and file extensions are restricted to three characters. A header record is created for each file. The header includes the file name and extension in RADIX50 format (any path names are discarded). The date of last modification is also included along with an arbitrary user identification code (UIC) of (1,1) and a protection code of 233. This code allows any accesss by the owner. The user group and all others may read or execute but not delete or write the file. This is the default code provided by the DOS system.

-n is a number 0-3 specifying a tape drive. Drive 0 is the default.

FILES

/dev/rmt[0-3] - mag tape drives 0-3
/dev/rmt[4-7] - same drives with no rewind

BUGS

- requires a non-standard mag tape driver to be able to create multi-file output tapes.

- present version reads entire tape on input. Don't specify file names.

- changes may be necessary due to the handling of EOF status from the mag tape driver.

AUTHOR

Pat Dowd

NAME

macro - assembler for the MACRO-11 language under Unix

SYNOPSIS

macro [option1 option2 ...] file1 file2 ... filen

DESCRIPTION

Macro assembles the concatenation of the specified files, terminating when a ".end" statement is encountered. The resulting object file is named filen.obj. If a file arg does not contain a ".", file.m11 will be sought before file itself.

Options, if desired, may appear anywhere in the command, and are chosen from the following:

-ls	Produce an assembly listing named filen.lst
-cr	Produce a cross-reference listing named filen.lst. If the -ls option is also included, the cross reference listing will follow the assembly listing in filen.lst
-lp	Same as -ls, but spools filen.lst for printing and deletes it when complete.
-xs:n	Allocates nK words of extra space for symbol table and macro storage. n must be octal and less than 20
-i:file	Takes arguments from file file seperated by blanks, tabs, or nl
-no	No object file is produced. This is useful for syntax checking or producing the listing only.
-ns	No symbol output in object modules. Thus ddt will not know of symbols from this assembly
-sm:fname	System Macros will be read from file "./fname.sml" instead of the standard file /usr/lib/sysmac.sml
-li:options	.list switch options seperated by :. Similarly for the -nl:, -en:, and -ds: switches.

FILES

/usr/lib/sysmac.sml - system macro library (for .mcall use)
/bin/lpr - spooler
/usr/lib/macxrf - cross-reference post-processor
filen.xrf - cross-reference temporary file

SEE ALSO

DEC Macro-11 Manual

BUGS

- .enabl/.dsabl for REG and GLB are ignored.
- Psect attributes are different. RO and RW become SHR and PRV, and I, D, HGH, and LOW are not supported. Unix segment attributes INS, DAT, and BSS work in their own fashion, but are not treated correctly by other programs.
- Listing header and footer formats should be changed to official RTL FR formats.

BUGS FIXED

- switches for li nl fl en ds exist, yet do not work.
- Undiagnosed troubles with local symbols have been experienced.

NOTE ************* These features will not be supported
for use under BOS. All psects will be treated as initialized
FORMAT: .psect name at1, at2, ...

Where name is the name to be assigned the psect and the 'at's' are
the attributes of the psect. the attributes are chosen from these:

WARNING ***** only initialized data segments will be supported
under BOS, with all code being one chunk *******

1. CON - concatenated - successive instances of this named psect will be linked head-to-tail.
2. OVR - overlay - successive instances of this named psect will overlay each other, such that the length of the psect will be dictated by the longest single instance.
3. LCL - local - the scope of this named psect is limited to this assembly.
4. GBL - global - the scope of this psect is global to all object modules being linked.
5. REL - relocated - the contents of this named psect are to be relocated at link time.
6. ABS - absolute - the contents are absolute.
7. SHR - shared - the contents of this psect are to be shared & read-only.
8. PRV - the contents are to be private & read/write.
9. BSS - this psect actually has no contents, but the addresses within it will be initially zeroed when the program is executed. BSS should be used for all data which need not have specific contents initially, as it will not have to be read in from disk. Any data generated within a BSS psect will merely reserve space, hence, the only useful directives are .BLKW & .BLKB

The default attributes are: OVR, GBL, REL, PRV
The attributes need be given only for the first mention of a particular named psect within an assembly. The first 8 attributes above are actually complementary pairs, while BSS carries the implications of PRV.

8

NAME

linkr - linking loader for Macro-11 under Unix

SYNOPSIS

linkr [option1 option2 ...] file1 file2 ... filen

DESCRIPTION

Linkr link-loads the specified files in the order given, producing an executable file named filen.out. If a file arg does not contain a ".", file.obj will be sought before file itself.

Options, if desired, may appear anywhere in the command, and are chosen from the following:

-ls	Produce a load map named filen.map
-cr	Produce a global cross-reference listing named filen.map. If the -ls option is also included, the cross-reference listing will follow the load map in filen.map.
-lp	Same as -ls, but spools filen.map for printing upon completion.
-no	No .out file is produced. This is useful for syntax checking or producing the listing only.
-xs:n	Allocates nK words of extra space for symbol table. n must be octal and less than 20.
-ns	no symbol output with load module. Thus .out file is "stripped"
-go	only global symbols are output in the .out file
-t:n	top load with next free byte after program being n(octal).
-b:n	bottom load with load point starting at n(octal).
-dc	DOS compatible option to force contiguous blank csect

FILES

filen.xrf - intermediate cross-reference temporary file
/bin/lpr - spooler
/usr/lib/macxrf - cross-reference post-processor

SEE ALSO

DEC Linkr-11 Manual

BUGS

- No overlay capabilities exist
- No transfer address switch exists
- Order of psect allocation is in order encountered, unlike DOS 9 which puts them in alphabetical order. The -dc switch does only part of the job.
- No relocation information is produced
- No support of ddt or odt is provided
- No library capability exists

NAME

lmc - load module conversion between Unix and DOS

SYNOPSIS

```
lmc
            file1   file2
```

DESCRIPTION

The first file is expected to be either a DOS or Unix load module, and is converted to a load module of the other type, becoming file2.

Only Unix type 407 files are acceptable as input or produced as output. No relocation or symbol table will be included in a Unix type output, ie the file will be 'stripped'.

SEE ALSO

a.out(V), piptp(VI)

BUGS

- It should be possible to convert type 410 files.
- No .ident or odt transfer address is put in the DOS communication directory.
- The Unix file is not in a canonical form. This may cause problems in comparisons.

NAME: RLOAD - REMOTE LOADER FOR DOS LOAD MODULES.

SYNOPSIS -
RLOAD FILENAME LINE SPEED TEL. NO.

DESCRIPTION
RLOAD WILL OPTIONALLY ESTABLISH A DIALUP CONNECTION TO A REMOTE MINICOMPUTER AND TRANSMITT A DOS LOAD MODULE TO THE MINICOMPUTER. VARIOUS DIAGNOSTICS WILL BE PRINTED IF AN ERROR OCCURES.

ARGUMENTS

FILENAME - THE UNIX NAME OF THE FILE TO BE TRANSMITTED. IF RLOAD IS INVOKED FROM A DIRECTORY OTHER THAN THE DIRECTORY CONTAINING THE FILE TO BE TRANSMITTED, THE FULL DIRECTORY PATHNAME MUST BE PREFIXED TO THE FILENAME.

LINE - THE UNIX DEVICE LINE NUMBER TO BE USED FOR TRANSMITTING THE FILE. THE DIRECT LINK ON SYSTEM BLUE TO THE STAND ALONE PDP10 IN THE CIRCUIT LAB IS ON LINE 24. THE AUTOMATIC CALL UNIT, TO BE USED WITH THE TELEPHONE NUMBER OPTION, IS LOCATED ON LINE 00.

SPEED - THE BAUD RATE AT WHICH THE DATA TRANSMISSION SHOULD TAKE PLACE. FOR TELEPHONE CONNECTIONS, A 300 IS NEEDED. FOR THE DEDICATED DATA LINK TO THE 'STAND ALONE' PDP10 A 1800 BAD RATE SHOULD BE USED.

TEL. NO. - THE COMPLETE TELEPHONE NUMBER TO BE USED BY THE DIALUP CONNECTION. FOR SYSTEM BLUE OF DEPT 5161, THIS IS: 9 1+ ANC NXX ABCD.

EXAMPLE - FOR A DIALUP DATA LINK CONNECTION:
```
%RLOAD BLAH.LDA 00 300 919495330
```
FOR SYSTEM BLUE DEDICATED DATA LINK:
```
%RLOAD BLAH.LDA 24 1800
```

FILES - /USR/BIN/RLOAD

%