SOFTWARE—PRACTICE AND EXPERIENCE, VOL. 5, 295-308 (1975)

Hints on Distributing Portable Software

W. M. WAITE
Department of Electrical Engineering, University of Colorado, Boulder, Colorado 80302, U.S.A.

SUMMARY

Would-be implementors of portable software often encounter frustrating difficulties when attempting to enter the distributed text into their computers. This paper summarizes various problems which may arise, and indicates how the distributor can head them off. The conventions used by several successful distributors are given to illustrate possible solutions.

KEY WORDS Portability Distribution Software engineering Character sets

INTRODUCTION

Benjamin Franklin's maxim 'a little neglect may breed mischief' is clearly illustrated by the comment which I have often heard about portable software: 'We had no trouble whatever getting it to run... once we got it into the computer,' All too often the shear difficulty of 'getting it into the computer' causes a potential user to abandon a system in frustration and embark upon an equivalent development of his own. Such problems can be largely avoided if the original implementor devotes a modest amount of thought to the distribution of his software.

Portable software must, by its very nature, be transferred between machines with very different peripheral complements and operating systems. The major stumbling blocks fall into three broad categories:

Character set
Medium
Maintenance

In subsequent sections I shall explore the sources of these problems, and present solutions which have been employed by several distributors of software. I shall concern myself solely with the barriers to *effective distribution*; questions of portability and adaptability of the software itself have been discussed at length elsewhere.[1-5]

CHARACTER SET

Choice of a character set must be made at the very beginning of the project, before any code is written. (In some cases, this choice is a part of a larger decision to employ a particular programming language.) There is a tradeoff involved: A restricted character set requires awkward constructs (e.g. using '.GT.' instead of '>'), while a broad one will be unacceptable to some computers.

Received 20 November 1974

ASCII and its subsets

The only character set to receive formal approval from any standards organization is ASCII.[6] ISO has also approved a code which is identical to ASCII, except that interpretations of certain characters are purposely omitted to allow for 'national variants'. Only 95 of the 128 distinct ASCII characters have been defined as graphics and assigned printable representations; the remaining 33 are interpreted as *control characters*. Except in certain cases which I shall mention later, the distributor of portable software is concerned only with the graphics in his character set.

There are five recognized subsets of ASCII, containing 63, 64, 66, 89 and 95 characters respectively. In each case, all characters of the subset are interpreted as graphics. Figure 1 gives the definitions of each of these subsets.

```
0 1 2 3 4 5 6 7 8 9
A B C D E F G H I J K L M N O P Q R S T U V W X Y Z
+ - * / =
( ) [ ] < > ¬
, . ; : ? ! " '
@ # _ $ & %
space
```

(a) 63-character subset

characters of (a), except [and]
{ } \

(b) 64-character subset

characters of (a)
{ } \ (c) 66-character subset

characters of (a)
a b c d e f g h i j k l m n o p q r s t u v w x y z

(d) 89-character subset

characters of (d)
{ } \ | ` ~

(e) 95-character subset

Figure 1. *ASCII subsets*

Hollerith, BCDIC and EBCDIC

The earliest punched card code in general use was the Hollerith code, which provided for 48 graphics and no control characters. There was almost complete uniformity in the meaning assigned to 37 of these (space, 0–9, A–Z), but the remaining 11 varied somewhat. Two particular interpretation, the so-called 'A' (commercial) and 'H' (FORTRAN) character sets, were commonly used by IBM and became *de facto* standards (Figure 2(a)).

When early 7-track magnetic tape drives were being developed, reliability considerations dictated that each tape character be represented by at least two bits. This led to the use of an even-parity representation for each character, and hence a 63-character set. IBM adopted such a system, and most other manufacturers followed suit. The particular character set included both graphics and control characters, and finally became known as *binary-coded decimal interchange code* (BCDIC). Unfortunately, there was a wide variation in the graphics assigned to the BCDIC characters which lay outside of the Hollerith subset. Figure 2(b) gives the set of graphics listed in the IBM 1401 manual.[7]

There were many complaints about the size of BCDIC, and IBM introduced Extended BCDIC with System/360. Since EBCDIC was never standardized, it is difficult to know exactly what it contains. Figure 2(c) is derived from the notorious 'green card'[8] which we all memorized in the mid- and late sixties; I have seen other specifications more nearly identical to ASCII.

Characters common to the 'A' and 'H' sets:

0 1 2 3 4 5 6 7 8 9
A B C D E F G H I J K L M N O P Q R S T U V W X Y Z
− * /
, .
$
space

Characters existing only in the 'A' set:

& # % □ @

Equivalent characters from the 'H' set:

+ = ( ) '

(a) Hollerith (48-character set)

Characters from (a) (either set—48 characters)

< >
[ ]
; : ? !
\ ≠ ≢ ≠ ∨ Δ √

(b) BCDIC (63-character set)

Characters from (a), except □ (both sets—52 characters)

a b c d e f g h i j k l m n o p q r s t u v w x y z
< > ¬
; : ? ! "
_ ¢ |

(c) EBCDIC (89-character set)

Figure 2. Other common character sets

MEDIA

Choice of a medium for distributing the text depends primarily upon the target computer: The recipient must have peripheral devices capable of reading the transmitted information. In most cases the peripheral complement of the target machine will accept several media, and other criteria can be used:

Volume and weight
Durability
Capacity
Convenience

Volume and weight obviously affect the cost of shipping or carrying the information from one installation to another. Durability refers both to the medium itself and to the information recorded upon it. Capacity reflects the size of the character set which can be expressed using the medium, not the total amount of information which the medium can hold. Convenience is a measure of the amount of effort needed to record and retrieve the

text. This last is very difficult to quantify, involving the peculiarities of the operating system and the experience of both the distributor and recipient.

Listing

A simple program listing is the obvious medium for transmission of short algorithms. It is small and light, durable and has a very high capacity (there is virtually no limit to the character set which can be used). As long as the transmitted text is short, a listing rates high on convenience: The user merely enters it directly, performing any necessary transformation in the process.

Listings are used successfully in such situations as the ACM Algorithms (although machine-readable text is also available for some of these). They are also useful in describing bootstrap programs such as Figures 5 and 6 of this paper. Once the length of a program exceeds a page or so, however, the tedious and error-prone nature of the transcription process begins to outweigh the advantages of the listing.

Cards

Punched cards of 80 columns[9,10] are a reasonable medium for transmitting programs ranging up to several thousand lines, although the weight and volume become serious factors beyond about 2,000 lines (one box). They are relatively durable if properly packed, but may be rendered useless by high humidity and careless handling. Probably the most serious threat to the integrity of the information comes at the recipient's installation, where the cards can be shuffled or mangled by either the operator or the computer. A distribution deck should *always* be serialized, and should be accompanied by a listing which can be used to verify the text once it is in the target computer.

Hole combinations are defined[11] for all 128 ASCII characters, plus an additional 128, but many card readers are not capable of reading the full set. Virtually any installation can read the 48-character Hollerith subset (Figure 3(a)), and most will handle the extension to 64 characters shown in Figure 3(b). (In some cases the hole combinations 11–0 and 12–0 are substituted for 11–8–2 and 12–8–2.) If a computer accepts binary cards then its card reader is, of course, capable of reading any hole combination. Unfortunately, it is often difficult to convince the operating system to accept cards with arbitrary punches which do not conform to the 'standard' format of a binary card.

Blank	1	2	3	4	5	6	7	8	9
12	12–1	12–2	12–3	12–4	12–5	12–6	12–7	12–8	12–9
11	11–1	11–2	11–3	11–4	11–5	11–6	11–7	11–8	11–9
0	0–1	0–2	0–3	0–4	0–5	0–6	0–7	0–8	0–9
	8–3		8–4						
	12–8–3		12–8–4						
	11–8–3		11–8–4						
	0–8–3		0–8–4						

(a) 48-character Hollerith card code

Codes from (a)

8–2	8–5	8–6	8–7
12–8–2	12–8–5	12–8–6	12–8–7
11–8–2	11–8–5	11–8–6	11–8–7
0–8–2	0–8–5	0–8–6	0–8–7

(b) 64-character BCD card code

Figure 3. Punched card codes

Cards are generally convenient to produce and to enter into the target computer; for this reason recipients are often willing to pay the extra cost of shipping them. Sometimes, however, conventions used by the target computer's job control language get in the way. For example, one operating system changes the character set when it sees a particular character in column 1; another treats /* in columns 1 and 2 as an end-of-file indication on cards. There is very little that the distributor can do to avoid these problems, but the recipient can usually get round them rather easily once he has identified the source of the difficulty.

Paper tape

Paper tape[12] is most useful in the same range of program size as cards—up to several thousand lines. It normally holds 10 characters per inch of tape, and lines are terminated by a control character. The packing density is thus better than that of cards because trailing spaces can be eliminated from each line. Paper tape is less susceptible to damage in transit than cards, and lines cannot easily be interchanged by the operator or reader.

Hole combinations are defined[13] for all 128 ASCII characters, and most modern tape equipment will handle this representation. Paper tape is convenient to produce and enter into most minicomputers and time-sharing systems, although it may require excessive amounts of time if no high-speed paper tape peripherals are available.

Magnetic tape

Magnetic tape[14] comes into prominence for longer texts, with a single 2,400-ft reel capable of holding from 10^5 to 10^6 lines (depending upon the recording density). Shorter text can be fitted onto smaller reels, with consequent savings in weight. Shipping costs for magnetic tape will probably be lower than those for any other medium once the size of the text exceeds a few hundred lines.

Since it is a magnetic medium, the information may be subject to corruption by certain kinds of baggage inspection devices. I have heard sad tales of loss attributed to x-rays, airborne weather radar and airport security devices; personally, I have never lost information on tapes which I have carried round the world, nor those which I have sent by air or surface mail. Customs inspectors are another matter, however: In Sydney, an official was reluctant to let me go without playing the tape for him (they are very wary of pornography—I calmed him by showing a listing); an inspector in Asmara wanted to unroll it to make sure that there was no movie film on the reel! When magnetic tape is sent by mail, the recipient may be charged duty unless he can prove that he furnished the tape initially.

Information may be recorded in either 7 or 9 *tracks* along the tape. Standard 9-track representations are defined[15-17] for all 128 ASCII characters, plus an additional 128. There is no standard 7-track representation; many systems support a 63- or 64-character subset, depending upon whether even or odd parity is used. It is also common practice to encode three 8-bit characters as four 6-bit characters on a 7-track tape. This permits representation of the full character set, but requires that the tape be written with odd parity.

Magnetic tapes are probably the *least* convenient of the media discussed so far. The problem is that every operating system seems to have its own idea of the proper format for the recorded information. If the recipient makes a specific request, then the distributor must convince his operating system to produce that format; generally an independent verification of his success (beyond the howls of the recipient) is not possible. When the recipient does not specify the format, or when the distributor himself tucks a tape under his arm and sallies forth to conquer new installations, a format must be chosen to fit the widest variety of situations.

Some operating systems require descriptive labels[18] to be recorded on the tape with the information. These labels define the format of the information, the names of the files, the dates on which they were created and so forth. Experience has shown that it is best to record these labels unless the recipient specifically requests an unlabelled tape: When a system knows nothing about labels, they are simply extra files to be skipped; if the system requires them and there are none, getting the tape read may be a virtually impossible task.

Each line of text constitutes a *logical record*, while each group of characters on tape constitutes a *block*. The tape labelling conventions of Reference 18 allow the distributor to group records into blocks such that:

1. No explicit indication of the boundaries between records is required.
2. There shall be an integral number of records per block.
3. Any use of padding requires the agreement of the interchange parties.
4. Truncated blocks are permissible.

The length of each block can be neither less than 18 characters[15-17] nor greater than 2,048 characters.[15-18] (This upper limit can be relaxed by agreement between distributor and recipient.) Record length may be *fixed* (all logical records in the file are of the same length), *variable* (each record contains its length in the first four character positions) or *undefined* (the applicable conventions have been agreed upon by the interchanging parties).

Two considerations are important in determining the block size to be used in the absence of particular specifications: the information density and the effects of a mismatch with the target computer's word length. Let us assume that we are planning to record fixed-length, 80-character records at 800 cpi. Reference 16 tells us that the nominal length of the inter-block gap is 0·6 in. If each block contains only a single logical record, then it will be 0·1 in long and 1/7 of the tape will contain useful information. On the other hand, if each block contained 25 records it would be 2·5 in long and 25/31 of the tape would contain useful information. Higher blocking factors increase the tape utilization, but must be agreed upon by the interchanging parties.

When a computer reads a block that does not fill an integral number of words, it may either truncate the block or fill the remainder of the last word with some padding character. To avoid confusion, it is best to choose a block size which is likely to fill an integral number of words on most computers. Since the most common word lengths (in characters) seem to be 2, 4, 6, 8 and 10 a suitable block size would be any common multiple of these values. The largest such block size which does not require agreement between the interchanging parties is 1,920 characters (24 card images).

MAINTENANCE

Much as the distributor would like to deny it, errors do appear after software is sent out. In some cases the distributed software may still be under active development, and the distributor will wish to supply improvements without re-distributing the entire package. Effective maintenance requires careful planning before the software is actually sent out, since the text must contain additional information and additional tools must be provided.

When the distributor updates his own master file, he should supply the commands which performed the update to all recipients of the affected software. In order to apply these commands mechanically, each recipient must have a text-editing program identical to that used by the distributor. Thus the distributor should supply a portable editor as a part of any package for which he intends to provide maintenance.

The portable editor should be designed to match the particular format used in the distributed text, and to provide only the facilities necessary for maintaining the software. It need be neither fancy nor particularly efficient, since it will be used only when the recipient wishes to introduce distributed changes. The recipient must be cautioned, however, to retain a copy of the text in a form suitable for input to the editor.

One basic question in the design of any text editor is how to identify the position at which a change should be made. Several methods are possible,[19] but in this case only one seems practical: Attach a line number permanently to each line. This guarantees that the changes supplied by the distributor are uniquely defined, regardless of additional changes which may have been made by the recipient.

Permanent attachment of line numbers does have drawbacks, however. Since the deck is not re-serialized after each update, it is more difficult to spot missing lines. Lines which do not appear may have been lost in transmission, or they may simply have been deleted by an earlier modification. Redundant transmission (such as including a listing, or recording several copies of the information on a tape) may distinguish these cases, but more should be done to give the recipient confidence in his implementation: Standard test data should be supplied with the software to validate its operation.[20] If this data produces extensive output, it is useful to include a machine-readable version of the output which can be compared mechanically with that generated by the new implementation.

EXAMPLES

In earlier sections, I have raised a number of points which should be considered when preparing portable software for distribution. This section will illustrate several sets of conventions which have been used by distributors of currently available software.

Mobile programming system

MPS is a collection of machine-independent software (distributed by the University of Colorado) for implementing processors *via* abstract machine modelling.[21] It includes the STAGE2 macro processor and processors for two list manipulation languages.[5]

Figure 4 shows the beginning of the first file of the distributed text for MPS. The 63-character subset of ASCII is chosen for compatibility with even-parity, 7-track magnetic tape. It is somewhat restrictive, precluding the use of lower case letters and a few special characters. Even with the use of a restricted set, however, experience has shown that automatic translation of character sets is an important facility to provide. The description therefore continues as shown in Figure 5, giving a simple program for performing this translation.

Figure 5 can also be used when the character set specification on the file does not begin in the first character position of the line. For example, when a program is being distributed as a single item, the character set may be given as an initial comment line. Often such lines must begin with some delimiter (such as the C in column 1 required by FORTRAN or the ALGOL keyword *comment*). As long as the delimiter plus character set is no longer than 72 positions, the program requires no alterations.

When the recipient does not specify a particular format, MPS is distributed on 7-track magnetic tape recorded at 800 characters per inch with even parity. The tape has standard labels, and each block of information contains 24 logical records. A logical record is a card image, serialized in columns 73–80.

Maintenance is provided by a simple editor called FIMP[22], which is written in ANSI FORTRAN and included in the distribution. A much-simplified version of FIMP is suggested for use in the initial processing of the distribution tape (see Figure 6, which is also an excerpt from the first file of the distribution tape).

```
0123456789ABCDEFGHIJKLMNOPQRSTUVWXYZ +-*/=(),.$'[]<>¬:;?!"&#%_@          CHAR0001
                                                                         CHAR0002
   THIS TAPE IS WRITTEN USING A 63-CHARACTER SUBSET OF THE AMERICAN      CHAR0003
NATIONAL STANDARD CODE FOR INFORMATION INTERCHANGE (ASCII).  FOR A       CHAR0004
COMPLETE DESCRIPTION OF ASCII, SEE AMERICAN NATIONAL STANDARDS X3.4-1968CHAR0005
AND X3.26-1970.  CHARACTER POSITIONS 1-63 OF THE FIRST LINE OF THIS      CHAR0006
FILE CONTAIN THE CHARACTERS OF THE SUBSET.  THEIR CONTENTS SHOULD BE     CHAR0007
THE FOLLOWING-                                                           CHAR0008
                                                                         CHAR0009
     POSITION      CHARACTER                                             CHAR0010
                                                                         CHAR0011
       1-10        DIGITS ZERO THROUGH 9                                 CHAR0012
      11-36        LETTERS A THROUGH Z                                   CHAR0013
         37        SPACE                                                 CHAR0014
         38        PLUS                                                  CHAR0015
         39        MINUS                                                 CHAR0016
         40        ASTERISK (MULTIPLICATION)                             CHAR0017
         41        SLASH (DIVISION, SOLIDUS)                             CHAR0018
         42        EQUAL                                                 CHAR0019
         43        LEFT PARENTHESIS                                      CHAR0020
         44        RIGHT PARENTHESIS                                     CHAR0021
         45        COMMA                                                 CHAR0022
         46        PERIOD (DECIMAL POINT)                                CHAR0023
         47        CURRENCY SYMBOL (DOLLAR)                              CHAR0024
         48        APOSTROPHE (SINGLE QUOTE, INVERTED COMMA)             CHAR0025
         49        LEFT SQUARE BRACKET                                   CHAR0026
         50        RIGHT SQUARE BRACKET                                  CHAR0027
         51        LESS THAN (LEFT ANGLE BRACKET)                        CHAR0028
         52        GREATER THAN (RIGHT ANGLE BRACKET)                    CHAR0029
         53        NEGATION                                              CHAR0030
         54        COLON                                                 CHAR0031
         55        SEMICOLON                                             CHAR0032
         56        INTERROGATION (QUERY, QUESTION MARK)                  CHAR0033
         57        EXCLAMATION POINT (FACTORIAL MARK)                    CHAR0034
         58        QUOTATION (DOUBLE QUOTE)                              CHAR0035
         59        AMPERSAND (AND SIGN)                                  CHAR0036
         60        SHARP (HASH MARK, NUMBER SIGN)                        CHAR0037
         61        PERCENT                                               CHAR0038
         62        UNDERLINE                                             CHAR0039
         63        AT RATE OF                                            CHAR0040
                                                                         CHAR0041
NOT ALL OF THE FILES ON THIS TAPE USE THE ENTIRE 63-CHARACTER SUBSET.    CHAR0042
MOST USE ONLY THE FIRST 48 CHARACTERS, WHICH CONSTITUTE THE IBM H SET    CHAR0043
DEFINED FOR FORTRAN.  IN SOME CASES CHARACTERS 49-60 WILL ALSO APPEAR    CHAR0044
IN A FILE, BUT CHARACTERS 61-63 ARE VERY RARE.  CONSULT THE              CHAR0045
DOCUMENTATION FOR EACH FILE (BELOW) TO DETERMINE THE SPECIFIC            CHAR0046
CHARACTERS USED.                                                         CHAR0047
```

Figure 4. MPS character set description

ALTRAN

ALTRAN is a programming system (distributed by Bell Telephone Laboratories) for performing symbolic computations on rational functions in several indeterminates with integer coefficients.[23] It consists of two major components, the ALTRAN Translator and

the ALTRAN Run-time Support Library. The basic system has been implemented in FORTRAN. The recipient may choose either of two character sets: EBCDIC or the Honeywell 6000 Series 6-bit character code.[24] Only 54 distinct characters are used in ALTRAN, and all but one (a hyphen) are available in the 63-character subset of ASCII. Since the hyphen is mapped into the EBCDIC underline, we can assume that this ASCII graphic would also be satisfactory. The first four lines of the distribution tape contain a complete character set, which is described in the installation manual.[24] Hence a program similar to that of Figure 5 could be used to provide automatic translation of the tape.

ALTRAN is distributed on magnetic tape written with odd parity, and the recipient may choose both the number of tracks and the density. Possible combinations (tracks/density) are 7/556, 7/800 and 9/800. The tape has no labels, and it is blocked with 20 logical records per block. Logical record size varies, but is at least 80 characters. The additional characters can be ignored by the recipient; they are 'artifacts of the hardware and software system used to produce the tape'.[24] Each logical record contains a single card image, serialized in columns 73–80.

```
   IF THE PRINTED GRAPHICS DO NOT CORRESPOND TO THE DEFINITION GIVEN     CHAR0049
ABOVE, THEN THE FOLLOWING ALGORITHM (EXPRESSED AS A FORTRAN PROGRAM)     CHAR0050
MAY BE USED TO PERFORM A SYSTEMATIC CONVERSION-                          CHAR0051
                                                                        CHAR0052
      DIMENSION IALF(72),ICHR(72)                                       CHAR0053
      COMMON LINE(80)                                                   CHAR0054
      READ (5,100) IALF,NALF,IUNIT,JUNIT,KUNIT                          CHAR0055
      CALL RDINT(IUNIT,JRES)                                            CHAR0056
      IF (JRES .NE. 0)  STOP                                            CHAR0057
      DO 1 I=1,NALF                                                     CHAR0058
      J=LINE(I)                                                         CHAR0059
    1 ICHR(J)=IALF(I)                                                   CHAR0060
    2 CALL RDINT(JUNIT,JRES)                                            CHAR0061
      IF (JRES .NE. 0)  STOP                                            CHAR0062
      DO 3 I=1,80                                                       CHAR0063
      J=LINE(I)                                                         CHAR0064
    3 LINE(I)=ICHR(J)                                                   CHAR0065
      WRITE (KUNIT,101) LINE                                            CHAR0066
      GO TO 2                                                           CHAR0067
  100 FORMAT(72A1,4I2)                                                  CHAR0068
  101 FORMAT(80A1)                                                      CHAR0069
      END                                                               CHAR0070
      SUBROUTINE RDINT(IN,JF)                                           CHAR0071
      COMMON LINE(80)                                                   CHAR0072
C        READ ONE LINE FROM UNIT IN.  IF AN ERROR OCCURS OR END-OF-FILE CHAR0073
C        IS SENSED, SET JF NONZERO AND RETURN.  OTHERWISE, PLACE AN     CHAR0074
C        INTEGER REPRESENTATION OF EACH CHARACTER OF THE LINE INTO THE  CHAR0075
C        CORRESPONDING ELEMENT OF THE COMMON ARRAY LINE, SET JF TO ZERO CHAR0076
C        AND RETURN.  THE MAPPING FROM CHARACTERS TO INTEGERS IS        CHAR0077
C        COMPLETELY ARBITRARY, SO LONG AS EACH CHARACTER MAPS INTO A    CHAR0078
C        UNIQUE INTEGER.                                                CHAR0079
      END                                                               CHAR0080
                                                                        CHAR0081
   TO USE THE CONVERSION PROGRAM, PUNCH A CARD CONTAINING THE PROPER    CHAR0082
CHARACTERS (OR ACCEPTABLE SUBSTITUTES) IN COLUMNS 1-NN, WHERE NN IS THE CHAR0083
SIZE OF THE SUBSET REQUIRED FOR THE DECK WHICH YOU WISH TO TRANSLATE.   CHAR0084
ALSO PUNCH THE NUMBER NN IN COLUMNS 73-74, AND THE FORTRAN UNIT NUMBERS CHAR0085
FOR THIS FILE, THE FILE TO BE CONVERTED, AND THE FILE ON WHICH THE      CHAR0086
RESULT IS TO BE WRITTEN IN 75-76, 77-78 AND 79-80 RESPECTIVELY.  IF     CHAR0087
CARD INPUT IS NOT FROM UNIT 5 AT YOUR INSTALLATION, ALTER THE FIRST     CHAR0088
READ STATEMENT ACCORDINGLY.                                             CHAR0089
```

Figure 5. Automatic character conversion

```
      DIMENSION INPUT(20),IDENT(2)                                      CONT0036
   10 READ (5,100) I,IUNIT,IDENT                                        CONT0037
      GO TO (1,1,3,4,5,6),I                                             CONT0038
    1 READ (IUNIT,101) INPUT                                            CONT0039
      IF (I .EQ. 2)  WRITE (JOUT,101) INPUT                             CONT0040
      IF (INPUT(20) .NE. IDENT(2))  GO TO 1                             CONT0041
      IF (INPUT(19) .NE. IDENT(1))  GO TO 1                             CONT0042
      GO TO 10                                                          CONT0043
    3 JOUT=IUNIT                                                        CONT0044
      GO TO 10                                                          CONT0045
    4 REWIND IUNIT                                                      CONT0046
      GO TO 10                                                          CONT0047
    5 STOP                                                              CONT0048
    6 ENDFILE IUNIT                                                     CONT0049
      GO TO 10                                                          CONT0050
  100 FORMAT(I1,I2,2A4)                                                 CONT0051
  101 FORMAT(20A4)                                                      CONT0052
      END                                                               CONT0053
                                                                        CONT0054
   COMMANDS DEFINING THE ACTIONS TO BE PERFORMED BY THE FILE UTILITY    CONT0055
ARE READ FROM UNIT 5.  (IF CARD INPUT IS NOT FROM UNIT 5 AT YOUR        CONT0056
INSTALLATION, ALTER THE FIRST READ STATEMENT ACCORDINGLY.)  EACH COMMANDCONT0057
IS PUNCHED IN COLUMNS 1-11 OF A CARD.  A DIGIT IN COLUMN 1 SPECIFIES THECONT0058
COMMAND, COLUMNS 2-3 CONTAIN A FORTRAN UNIT NUMBER (IF USED) AND COLUMNSCONT0059
4-11 CONTAIN A CARD IDENTIFIER (IF USED).  THE EFFECT OF EACH COMMAND ISCONT0060
DEFINED AS FOLLOWS-                                                     CONT0061
                                                                        CONT0062
          COMMAND                                EFFECT                 CONT0063
                                                                        CONT0064
1  FORTRAN UNIT  IDENTIFIER  SKIP CARDS ON THE GIVEN UNIT UNTIL A CARD   CONT0065
                             WITH THE GIVEN IDENTIFIER IS SKIPPED.      CONT0066
                             LEAVE THE GIVEN UNIT POSITIONED AFTER THE  CONT0067
                             CARD WITH THE GIVEN IDENTIFIER.            CONT0068
2  FORTRAN UNIT  IDENTIFIER  COPY CARDS FROM THE GIVEN UNIT TO THE       CONT0069
                             CURRENT OUTPUT UNIT UNTIL A CARD WITH THE  CONT0070
                             GIVEN IDENTIFIER IS COPIED.  LEAVE THE     CONT0071
                             GIVEN UNIT POSITIONED AFTER THE CARD WITH  CONT0072
                             THE GIVEN IDENTIFIER.                      CONT0073
3  FORTRAN UNIT              MAKE THE GIVEN UNIT THE CURRENT OUTPUT UNITCONT0074
                             FOR SUBSEQUENT COPY COMMANDS.              CONT0075
4  FORTRAN UNIT              REWIND THE GIVEN UNIT.                     CONT0076
5                            STOP                                       CONT0077
6  FORTRAN UNIT              ENDFILE THE GIVEN UNIT.                    CONT0078
                                                                        CONT0079
   NOTE THAT AN OUTPUT UNIT MUST BE SET (3) BEFORE THE FIRST COPY       CONT0080
COMMAND (2) IS GIVEN.  INFORMATION MAY BE COPIED DIRECTLY FROM THE FILE CONT0081
CONTAINING THE COMMANDS SIMPLY BY SPECIFYING ITS UNIT NUMBER IN A COPY  CONT0082
COMMAND.  CARDS FOLLOWING THE COMMAND WILL THEN BE COPIED, UP TO AND    CONT0083
INCLUDING THE ONE WHICH CONTAINS THE IDENTIFIER SPECIFIED BY THE COPY   CONT0084
COMMAND.  THE FOLLOWING CARD WILL THEN BE INTERPRETED AS A NEW COMMAND. CONT0085
(NO PARTICULAR FORMAT FOR AN IDENTIFIER IS ASSUMED BY THE PROGRAM, AND  CONT0086
HENCE ANY UNIQUE SEQUENCE OF EIGHT CHARACTERS MAY BE USED TO MARK THE   CONT0087
LAST CARD OF AN INSERTION FROM THE COMMAND FILE.)                       CONT0088
```

Figure 6. A simple file utility

Maintenance is provided by an editor[25] which is on the distribution tape. This editor is also used heavily during installation, and the installation manual contains an 11-line FORTRAN program used to extract the source text for the editor. (This program is much less general than that of Figure 6, since its only purpose is to obtain the editor source text.)

BCPL

BCPL is a programming language (distributed by the University of Cambridge) intended for implementation of system software.[26, 27] The distribution package includes two versions of a compiler which converts BCPL into a simple intermediate language: One version is written in BCPL, the other has been translated into the intermediate language. To run BCPL on his computer, the recipient must build a translator or interpreter for the intermediate language and then perform a standard bootstrapping sequence.[28]

The distributed text is in EBCDIC, but each character is encoded as a pair of hexadecimal digits. Thus only the characters 0–9 and A–F appear on the transmission medium—a very restricted character set indeed! Automatic character set translation is provided by the routine which converts the pairs of hexadecimal digits to selected graphics of the recipients' character set. Source lines are packed, using the EBCDIC control character NL (new line) to mark the end of each. This is an example of the use of control characters in addition to graphics, which I mentioned when discussing character sets.

The default format for the distribution tape is apparently 9-track, 800 characters per inch, odd parity. It is unlabelled, with 80-character EBCDIC records and a block size of 1,600 characters. The first 72 character positions of each record hold 36 pairs of hexadecimal digits, and thus encode 36 characters of the text. Character positions 73–80 contain a sequence number of up to 8 digits, right-adjusted and filled with spaces. Note that these sequence numbers bear no relationship whatever to the lines of source text; they simply provide a check on the completeness of the encoded representation.

There is no provision for distributing changes to BCPL.

SNOBOL4

SNOBOL4 is a programming language (distributed by Bell Telephone Laboratories) intended for processing of symbolic information.[29] The distribution package contains a compiler/interpreter for SNOBOL4, written in an abstract machine language called SIL.[30] SIL operations are written in the form taken by many assembly languages; it is intended that the recipient implement them as macros in his local assembly language.

The compiler/interpreter was developed on System/360, and hence EBCDIC characters are used. The large majority of the characters in the text belong to the Hollerith set, however, with additional characters appearing only in occasional commentary. The tape does not contain a complete list of the characters used, although such a list does appear in the implementation guide.[31] Thus any automatic character translation must be based upon the recipient's knowledge of the response of his system to the EBCDIC codes on the distribution tape.

When the recipient does not specify a format for the SIL source text, it is recorded with odd parity on a 9-track tape at 800 characters per inch. Standard labels are provided, and the information is written in 3,200-character blocks. Each logical record is an EBCDIC card image, serialized in columns 73–80.

An 'alter editor' is provided for incorporating distributed changes. This program is written in SNOBOL4, and hence the system must be running before it can be used. Although the original text is serialized, the alter editor determines line numbers by counting from the beginning of the deck. Thus a particular number is not invariably associated with a given line. Some care must therefore be taken to make the alterations in the proper order and to ensure that no local changes are made.

NATS

The National Activity to Test Software (NATS) is based at Argonne National Laboratory. Their concern is the distribution of highly reliable numeric packages written in FORTRAN, and hence their character set requirements are modest. They use either BCDIC or EBCDIC, depending upon the target computer. No automatic translation is provided.

Two formats are standard for NATS software: 7-track magnetic tape recorded at 556 characters per inch with even parity, and 9-track magnetic tape recorded at 800 characters per inch with odd parity. The 7-track tape is unlabelled, with unblocked 80-character records, and uses the BCDIC character set. A block size of 3,200 characters (40 card images) is used on the 9-track tape, which is written in EBCDIC.

NATS apparently makes no provision for distributing changes, although their documentation encourages users to report unsatisfactory performance.

ACM algorithms

Certain of the algorithms published in Communications of the ACM are made available on magnetic tape. Currently, only FORTRAN versions are distributed and hence the Hollerith character set is adequate. No information is included to permit automatic character translation.

The default tape format is identical to the 7-track NATS distribution: An unlabelled, even parity tape written in BCDIC at 556 characters per inch. Although each record is uniquely identified by the first three latters of the subroutine and the serial number of the card (incremented by 10 for each card), there is apparently no provision for maintenance of the distributed algorithms.

CONCLUSION

Several years ago, one of the minicomputer vendors ran an advertisement with the caption 'Our hardware is delivered—not abandoned'. Software distributors should keep this in mind: If software is to be delivered rather than abandoned, then considerable effort must be devoted to the details of distribution.

In choosing a distribution format, you must make every decision with a view to easing the recipient's task. I would advocate restricting the characters set to no more than 63 graphics (48 is even safer), providing a full list of the characters set as the first line of the text, using standard labels on magnetic tape, and including facilities for maintenance. Be prepared to provide a variety of formats at the request of the recipient, and to answer queries from recipients who are having implementation problems.

Extensive documentation is absolutely essential to successful distribution. Use the computer to maintain this documentation,[32-34] and update it when user feedback indicates that it is inadequate. (If the documentation maintenance system itself is portable,[34] then the recipient can easily modify the documentation to meet any special needs.) As an indication of the dimension of the problem, consider some of the systems quoted in the previous section: The STAGE2 Implementation Guide is 30 pages long; those for ALTRAN and SNOBOL4 require 118 and 176 pages respectively. Test data should also be extensive and carefully thought out. (The test input for the various steps of STAGE2 implementation accounts for 1,722 of the 3,663 lines in the distribution file). All of this information must be updated as you obtain feedback from users and make modifications to the software.

At this point, you may wonder whether the game is worth the candle. My only answer is that you may want to use someone else's software someday—why not set a good example?

REFERENCES

1. M. H. Halstead, *Machine-Independent Computer Programming*, Spartan Books, Washington, D.C., 1962.
2. M. I. Halpern, 'Machine independence: its technology and economics', *CACM*, **8**, 782–785 (1965).
3. P. J. Brown, 'Levels of language for portable software', *CACM*, **15**, 1059–1062 (1972).
4. P. C. Poole and W. M. Waite, 'Portability and adaptability', *Lecture Notes in Economics and Mathematical Systems*, **81**, 183–277 (1973).
5. W. M. Waite, *Implementing Software for Non-Numeric Applications*, Prentice-Hall, Englewood Cliffs, N.J., 1973.
6. *Code for Information Interchange*, X3.4–1968, American National Standards Institute, New York, 1968.
7. *IBM* 1401/1460 *System Operation Reference Manual*, A24–3067–0, International Business Machines Corp., Endicott, N.Y., 1963.
8. *IBM System* 360 *Reference Data*, X20–1703–5, International Business Machines Corp., White Plains, N.Y., 1965.
9. *General Purpose Paper Cards for Information Interchange*, X3.11–1969, American National Standards Institute, New York, 1969.
10. *Rectangular Holes in Twelve-Row Punched Cards*, X3.21–1967, American National Standards Institute, New York, 1973.
11. *Hollerith Punched Card Code*, X3.26–1970, American National Standards Institute, New York, 1970.
12. *Specifications for Properties of Unpunched Oiled Paper Perforator Tape*, X3.29–1971, American National Standards Institute, New York, 1971.
13. *Perforated Tape Code for Information Interchange*, X3.6–1965, American National Standards Institute, New York, 1965.
14. *Unrecorded Magnetic Tape for Information Interchange*, X3.40–1973, American National Standards Institute, New York, 1973.
15. *Recorded Magnetic Tape for Information Interchange* (200*CPI, NRZI*), X3.14–1973, American National Standards Institute, New York, 1973.
16. *Recorded Magnetic Tape for Information Interchange* (800*CPI, NRZI*), X3.22–1973, American National Standards Institute, New York, 1973.
17. *Recorded Magnetic Tape for Information Interchange* (1600*CPI, PE*), X3.39–1973, American National Standards Institute, New York, 1973.
18. *Magnetic Tape Labels for Information Interchange*, X3.27–1969, American National Standards Institute, New York, 1969.
19. S. R. Bourne, 'Design for a text editor', *Software—Practice and Experience*, **1**, 73–81 (1971).
20. W. M. Waite, 'Building a mobile programming system', *Computer J.* **13**, 28–31 (1970).
21. M. C. Newey, P. C. Poole and W. M. Waite, 'Abstract machine modelling to produce portable software—a review and evaluation', *Software—Practice and Experience*, **2**, 107–136 (1972).
22. W. M. Waite, *File Manipulation Program*, Department of Electrical Engineering, University of Colorado, 1972.
23. W. S. Brown, *ALTRAN User's Manual*, Bell Telephone Laboratories, Murray Hill, N.J., 1973.
24. A. D. Hall, *ALTRAN Installation and Maintenance*, Bell Telephone Laboratories, Murray Hill, N.J., 1972.
25. A. D. Hall, *SEDIT—A Source Program Editor*, Computing Science Technical Report No. 16, Bell Telephone Laboratories, Murray Hill, N.J., 1974.
26. M. Richards, 'BCPL: a tool for compiler writing and system programming', *Proc. AFIPS SJCC*, **34**, 557–566 (1969).
27. M. Richards, *The BCPL Reference Manual*, Memo 69/1, University Mathematical Laboratory, Cambridge, 1969.
28. M. Richards, 'The portability of the BCPL compiler', *Software—Practice and Experience*, **1**, 135–146 (1971).
29. R. E. Griswold, J. F. Poage and I. P. Polonsky, *The SNOBOL4 Programming Language*, 2nd ed., Prentice-Hall, Englewood Cliffs, N.J., 1971.
30. R. E. Griswold, *The Macro Implementation of SNOBOL4*, W. H. Freeman & Co., San Francisco, 1971.
31. R. E. Griswold, *A Guide to the Macro Implementation of SNOBOL4*, S4D8c, Bell Telephone Laboratories, Holmdel, N.J., 1970.

32. S. L. Reed, *TEXT*360, 360D 29.4.001, International Business Machines Corp., White Plains, N.Y., 1967.
33. G. M. Berns, 'Description of FORMAT, a text-processing program', *CACM*, **12**, 141–146 (1969).
34. W. M. Waite, *A Program for Document Maintenance and Production*, PB 223 163, National Technical Information Service, Springfield, Va., 1973.