A TRANSPORTABLE SYSTEM FOR MANAGEMENT AND EXCHANGE OF PROGRAMS AND OTHER TEXT

W. V. Snyder

Jet Propulsion Laboratory
4800 Oak Grove Drive
Pasadena, California 91103

ABSTRACT

Computer software is usually exchanged between different computer facilities via punched cards or magnetic tape. For up to about 1000 images, cards are cheaper and probably easier to deal with than tape. The primary problem with cards is the variety of punch codes. Frequently, one also has the minor nuisance of repunching cards damaged in transit. For larger amounts of data, tape is cheaper, but there are so many tape formats in use that the recipient frequently has trouble reading a tape, even if the format is simple and well defined. Occasionally, one has the problem of parity errors, which make a tape essentially worthless. When test data, modules in several languages, computer output or documentation are included, the lack of organization in the material can cause a substantial amount of unnecessary human labor.

This paper presents a system for exchanging information on tape, that allows information about the data to be included with the data. The system is designed for portability, but requires a few simple machine dependent modules. These modules are available for a variety of machines, and a bootstrapping procedure is provided. The system allows content selected reading of the tape, and a simple text editing facility is provided. Although the system recognizes 30 commands, information may be extracted from the tape by using as few as three commands. In addition to its use for information exchange, we expect the system to find use in maintaining large libraries of text.

THE MOTIVE FOR DEVELOPING THIS PROGRAM was the experience of receiving tapes from many correspondents. We dealt with most correspondents only once or twice. We received tapes written in every possible density, both parity modes, several character codes, and having a variety of block and record lengths. We see three solutions to this problem. Most computer centers have access to a program that can handle fixed length records, written in fixed length blocks, using a popular code such as ASCII or EBCDIC. When the characteristics of the medium were correctly provided, we had good success with a program of this type*. Unfortunately, this information was not always provided, and was sometimes incorrect. Another solution is for some organization to promulgate a standard for record lengths, block lengths, codes and parity modes. Then if such information is not provided, the standard is a reasonable guess. Neither approach can cope with disorganization of the data, or with parity errors. We chose therefore to write a transportable program to enforce a standard recording format, organize the data and provide for error recovery. This relieves the sender of the responsibility for sending information about character codes, record lengths and block lengths with the tape. He must, of course, still tell the receiver the tape density, and whether it is a seven- or nine-track tape. Since some binary numeric information is recorded, only odd parity tapes may be used with this program.

*We used two programs, known as BLOCK and UNBLOCK, written in Univac-1100 assembler language at the University of Maryland Computer Science Center.

RECORDING FORMAT

Most computer systems can deal with ASCII information in a natural way. In order to use nine-track tape conveniently, we represent the seven-bit ASCII code using eight bits, with the high-order bit zero. The program does not, however, enforce this convention rigidly. Certain information must be encoded in this way, but the textual information may be encoded in any way that may be represented by a string of eight-bit units. It is preferable that all information be encoded in some standard form, and we hope that all implementations of the program will use ASCII code for the textual information.

Some computers can read or write tapes containing blocks consisting of an integral number of words, and can read tape blocks of arbitrary length only with difficulty. For example, a tape containing blocks consisting of ten 80-character records could be read only with difficulty on a Univac-1100, which expects nine-track tapes to contain blocks consisting of a multiple of nine characters, and could not be written on a Univac-1100. We therefore selected a block size having factors of nine (for 36-bit words), fifteen (for 60-bit words) and four (for 32-bit words). These factors also guarantee that the block will be an integral number of words if it is written on a seven-track tape. The program uses data the same for seven- and nine-track tapes.

Since information may be recorded on magnetic tape in blocks of arbitrary length, separated by gaps of fixed length, one can use less space on the tape to record a given amount of data by writing longer, and therefore fewer blocks. We chose to write information in blocks of 7200 characters. This block size allows efficient use of space on tape, and usually fits into a minicomputer memory. A 180-character label is the first block written on every tape. Information in the label includes the block size. If the program does not fit in available memory, smaller blocks may be written. The program can read the smaller blocks automatically. This was required in one minicomputer implementation of the program. We recommend that all implementations retain the capability to read 7200 character blocks. Further conservation of space on the tape is achieved by compressing the data. To compress the data, consecutive occurrences of blanks (or another character if desired) are removed, and replaced with an encoded representation requiring less space. A compressed Fortran program usually occupies about one third the space otherwise required.

DATA MANAGEMENT FACILITY

Although the problem of dealing with variable and frequently uncertain physical characteristics of the transmission medium was irritating, the problem that consumed most of our time was the uniform lack of organization of the information on the tape. We received programs in several languages, subprograms with several test drivers, multiple versions of a program, test data, computer output and documentation, with no indication of what was to be done with the information. In such situations, much effort was spent organizing the information before it could be used. We therefore developed not only a program to read and write tape, but also a transportable data management system for textual information.

Our data management scheme consists of recording each program, subprogram, listing or data group as a separate module of text. Helpful information about the module is recorded with the module. The minimum information required with each module is a name. For more complete identification of the module, one may record the data type (language for modules that are programs), machine type, authors' names and addresses, and bibliographic references. To facilitate management of programs consisting of several modules, one may record the names of groups of which the module is a member, and keywords related to the module. To control changes to modules, a simple but flexible updating mechanism is provided, and the updating history is recorded. To record information that does not fall into any of the specified categories, one may include comments. We call this information control information. All control information is recorded with the text of the module. The text and control information can be examined and updated separately, but they remain together on the tape.

A data management system requires a command language. In specifying the command language for the Exchange Program, our goals were simplicity and comprehensive flexibility. The use of the program is oriented primarily toward the receiver of the tape. Although the program acts on 30 commands, information may be extracted from the tape with as few as three commands:

```
    INTAPE = Fortran unit number of input tape
    OUTPUT = Fortran unit number of native format
             file
    COPY   = List of module numbers.
```

To create a new tape requires, at a minimum, the following commands:

```
    TITLE  = Title of tape
    SITE   = Site at which the tape is being
             written
    OUTAPE = Fortran unit number of the tape
    DATE   = Date written (YYMMDD) [May be provided
             automatically.]
```

Each module of the text must then be preceeded by

```
    INSERT = Name of module
    TEXT
```

and followed by an end of text signal. If more information about the module than its name is to be provided, more commands are required.

ERROR DETECTION AND CORRECTION

The program currently contains two error detection mechanisms. First, it uses the error detection mechanism of the supporting operating system. Second, it records a sequence number in each block, and checks it during reading. It also records in each block the location of the first record that starts in the block, the number of the text module of which it is a member, and the location of the first record of the first text module that begins in the block, if any. We plan to use this information for partial error recovery. We also plan a more ambitious error control algorithm, capable of detecting and correcting up to 72 consecutive erroneous characters, at up to four different places in each block. It can be implemented in a transportable way, requiring only a machine sensitive exclusive-or primitive operation. For the 7200 character block chosen as the standard for the Exchange Program, only 113 characters of error control information are required. The design of the block format includes provision for this information.

EXPERIENCE

The program has been used at JPL to manage the collection of algorithms submitted to ACM TOMS, for weekly exchange of data between a DEC PDP-11/55 and a Univac-1108, and occasional exchange of data between a Univac 1108, Sperry (formerly Varian) 72, and a DEC PDP-11/55. The program was used to transmit the JPL mathematics library to a DEC PDP-10 at the California Institute of Technology, and is currently used there to retrieve modules of the JPL mathematics library from the exchange tape. It was also used to transmit information to a CDC-6600 at Sandia Laboratories. Experience in implementing the program on the DEC PDP-11/55 and on the DEC PDP-10 indicated that changes in the interface between the portable and non-portable parts of the program are desirable. In particular, the DEC Fortran environment requires that data files be explicitly opened (with a non-portable statement) before they are used. Thus, a subprogram thought to be portable does not work on DEC machines. We expect to change the interface between the portable and non-portable parts of the program to concentrate potentially non-portable requirements in fewer places. When we make that change, we will also add a few commands.

SUMMARY

We have developed a transportable program for exchange of textual information that provides several advantages over previous methods. The program enforces the use of a standard tape format, uses tape efficiently, organizes the information on the tape, provides for simple retrieval of information from the tape, and provides for error recovery. Since the program is transportable, it is used similarly on all computer systems. Thus, once one learns to use the program, one may use the program on many computer systems with little additional effort.

ACKNOWLEDGEMENTS

The author thanks the following (in alphabetical order) for their help: Julian Gomez of JPL implemented the program on a DEC PDP-11 using RSX-11. Jeff Greif at the California Institute of Technology implemented the program on a DEC PDP-10. Dick Hanson and Karen Haskell at Sandia Laboratories in Albuquerque wrote some of the IBM system 360 modules. Karen Haskell implemented the program on a CDC-6600. Fred Krogh of JPL provided most of the requirements and several ideas for the command language. Robert McEliece of JPL provided an error detection code. West Coast University in Los Angeles provided free computer time to implement the program on a Varian 72.

This paper presents the results of one phase of research carried out at the Jet Propulsion Laboratory, California Institute of Technology, under contract No. NAS 7-100, sponsored by the National Aeronautics and Space Administration.