A programmer's utility filing system

By M. V. Wilkes*

A modern trend in computer organization is to provide facilities for a user to keep on magnetic tape or on a disc file all the information that he is likely to require in the immediate future. This includes current programs, old programs likely to be required again, and data of all kinds. PUFS is a system by which a user can create, edit, and up-date files of information which are stored on a magnetic medium in a form which is an exact image of the form in which they are written on paper.

The object of the Programmer's Utility Filing System (PUFS) is to provide the programmer with a means of storing on magnetic tape, or on a disc file, all the programs and data that he is making use of in his current work, and to reduce to a minimum the quantity of punched paper tape, or punched cards, that is used. The system is inspired by certain features of the Compatible Time-Sharing System (CTSS) in use experimentally at M.I.T., although the M.I.T. system, of course, also gives the user very much more, namely, direct access to the computer (Corbató *et al.*, 1963). PUFS has been implemented for EDSAC 2, and it is this implementation which will be described.

As in CTSS, the full flexibility of working to which the programmer is normally accustomed, is retained by arranging for the storage of information within the computer in the form of an exact image of that in which it is normally written or printed. In particular, one can store programs written in any programming language, and the system is to be distinguished from systems designed for the up-dating of programs written in particular languages. In PUFS, programs and data are originally punched on paper tape, but are then stored on magnetic tape in the same character-by-character form in which they appear when the input tape is printed on a teleprinter, except for the presence of line numbers, which do not necessarily exist in the original punched form. Once information has been read into the machine and converted into a file on magnetic tape, it can be used repeatedly by programs as though it were being read in for the first time, and editing of the sort that normally requires the preparation of up-dated paper tapes can be carried out wholly within the computer, only new information being punched.

PUFS is operated by a series of commands, which are read from the input tape under the control of the system, and then executed. A number of commands may be punched on the same tape and will be executed in sequence.

An INPUT command must be followed by a punched version of the document that it is desired to take into the machine. Normally, the lines of the document in its original form are unnumbered, and PUFS assigns numbers to them on input; if desired, however, some or all of the lines may have line numbers punched along with them, and in this case these line numbers take precedence. When input is complete, the lines are sorted into increasing numerical order of line numbers. If two lines have the same number, only that which had its number punched along with it is retained. If the punch operator has given the same number to more than one line, all but the last to be read into the machine are discarded. These provisions enable errors and omissions made during the punching of a document to be rectified without altering what has already been punched. Blank lines, that is, lines with a number but no information, are removed altogether, and this provides a means of deleting unwanted lines.

The line numbers also enable corrections or modifications to be made to a document already stored on magnetic tape. In order to make insertions possible without renumbering, the line numbers, when first introduced, start at 10 and go up in steps of 10. A MODIFY command is followed by the additional or corrected lines that are to be incorporated into the file. These lines must all be numbered and any lines accidentally left unnumbered will be ignored. On input to the computer the lines are arranged in numerical order and the redundant lines eliminated as for a primary document. The lines from the modification tape are then sorted in order with those of the file; where lines with the same numbers exist in both the file and the modification tape, those from the latter replace those in the former. As in the case of primary input, blank lines are eliminated.

Commands are provided for splitting files into two, and for joining two files together to make one. In the latter case, re-numbering of the combined file takes place automatically. Re-numbering can also be initiated as a special operation when desired. Files may be printed, either as they stand with line numbers, or with line numbers suppressed. The former will naturally be preferred when further editing has to be done.

An EXECUTE command enables assembly-language programs stored on magnetic tape to be run exactly as

* *Director, University Mathematical Laboratory, Corn Exchange St., Cambridge.*

they would be using direct input from paper tape. As a particular case, the program activated by an EXECUTE command may be a compiler for a higher level language, thus making it possible to use PUFS for programs written in that language. An EXECUTE command is followed by a list of files containing the program and data on which it must operate. Both program and data may, if necessary, extend over several files. It should be noted that an EXECUTE command causes control to pass from the PUFS system to the program to be executed and that, as at present implemented, the PUFS system has no way of regaining control. Control, may, however, be sent back to the PUFS system by orders placed in the program.

The magnetic-tape filing system

PUFS makes use of two magnetic tapes. Tape 1 is a system tape on which is recorded an executive subroutine, and various permanent files such as assembly routines and compilers. Tape 2 holds all the other files that the user has created, and the file directory. The file directory gives for each file a tape number and the number of the block on that tape at which its start is to be found.* File numbers are allocated by the user and lie in the range 1 to 2,047. Whenever a new file is added, it is written on the tape after the files already existing, and its number is added to the directory, along with the number of the block in which it starts. The same thing happens when a file is modified by the process described above; the old file is not erased, but the modified version is added to the collection of files as though it were a new file altogether. As a result, the file directory may contain several entries for a given file number. If the user refers to a file simply by its number, e.g. 27, the executive routine will automatically select the latest version; the user may, however, refer to earlier versions, if he wishes, by writing, for example, 27 1 or 27 3, which refer, respectively, to the last but one and the last but three versions of file number 27. Either the latest version of a file or a specified earlier version may, at the user's pleasure, be given a new number.

The user may, at any time, edit his file directory by deleting references to specified files or versions of them; whenever this is done, references to all earlier versions of the same file are automatically deleted as well. Note that the editing operation does not itself erase the unwanted files from the magnetic tape.

During the operations that have been described above, the executive routine and the file directory are held in the high-speed store. Since some of the operations bring about alterations in the file directory, it is necessary that, when all the operations have been performed, the up-to-date version of the file directory should be recorded on the file tape. This may be brought about explicitly by the use of a DUMP command, or, automatically, by an EXECUTE command. In both cases, the number of the block on the magnetic tape at which the file directory is dumped is printed out for the information of the user. The executive routine keeps account of all dumps that are made and prints this count in front of the block number. This is intended to be an aid to the programmer in keeping track of his output sheets, and may be regarded as a substitute for the printing of the date and time, which is not possible on EDSAC 2 since there is no clock.

The file tape may contain a number of file directories that have been dumped at different times. Normally the user will wish to restart with the latest file directory, and it is arranged that the block number at which this is to be found is recorded in a block reserved for that purpose at the beginning of the system tape. This is the only use—and not an essential one—made in PUFS of the facility available in EDSAC 2 for re-using an individual block on the magnetic tape. If the user wishes to go back to an old file directory, he may do this by making use of the knowledge available to him of where that directory is stored.

From time to time a user may ask for a fresh tape to be prepared by copying the old one. When this is done only files, or versions of files, whose numbers occur in the most recent version of the file directory are copied. Copying thus provides a way of dropping from the system files that are no longer required. If the old tapes are kept it is always possible to recover a former situation if this proves necessary, for example, on account of machine malfunctioning. In the present version of the system, copying of tapes is done by a special program and not by means of a PUFS command.

The commands

A PUFS command causes one of the following actions to take place:

(1) A file to be loaded into memory from paper tape or magnetic tape.
(2) A file loaded in memory to be given a number and copied on to magnetic tape.
(3) The file loaded in memory to have an operation performed on it (e.g. a modification or splitting).
(4) An alteration to be made in the file directory.

The commands that have been implemented so far are as follows:

INPUT. /	Load a file into memory from paper tape with corrections incorporated; the file is punched immediately after the command. Line numbers are indicated by having a special symbol punched in front of them (in the EDSAC 2 implementation a suffix $_{10}$ is used).

* In the EDSAC 2 magnetic-tape system, block marks and block numbers are pre-recorded on the tapes. Each block holds up to 50 words, but the routine used for recording information automatically makes use of as many blocks as are required. Thus a tape constitutes an addressable memory.

LOAD. *N i* /	Load file *N* (version *i*) from magnetic tape. If the file is not in the file directory the report FNF (file not found) is printed.
FILE. *N* /	Copy the loaded file on to magnetic tape and make the appropriate entry in the file directory.
PRINT. *n m* /	Print the loaded file starting at line number *n* and finishing with line number *m*. If *m* is omitted, printing starts at *n* and continues to the end, and if both *n* and *m* are omitted the whole file is printed.
PRINT SUPPRESSED. *n m* /	As PRINT but omitting line numbers.
MODIFY. /	Incorporate in the loaded file the corrections punched immediately after the command. A copy of the corrections, arranged in order of line numbers, is printed for record purposes.
APPEND. *N i* /	Attach the file *N* (version *i*) to the end of the loaded file and re-number the augmented file thus obtained.
PREFIX. *N i* /	Attach file *N* (version *i*) to the front of the loaded file and re-number the augmented file thus obtained.
DETACH TAIL. *n* /	Shorten the loaded file by removing all lines from *n* onwards (the tail). The tail ceases to be part of the loaded file, but is preserved for use if required.
USE TAIL. /	Load the tail left over from a previous DETACH TAIL operation. The file previously loaded is lost. If there is no tail the report NT (no tail) is printed.
RENUMBER. /	Re-number in sequence the lines of the loaded file, starting with number 10 and going up in steps of 10.
DELETE. *N i M j* . . . /	Remove from the file directory file *N* (version *i*), file *M* (version *j*), etc. All earlier versions of deleted files are also deleted.
PRINT FILE LIST. /	Print the file directory.
RENAME. *N i M* /	Gives the new name *M* to file *N* (version *i*). If the file cannot be found the report FNF is given.
DUMP. /	Dump the PUFS file directory. After execution the serial number of the dump and the number of the first block in which it takes place are printed.
EXECUTE. N_1 i_1 N_2 $i_2 \ldots M_1 j_1 M_2 j_2 \ldots$ /	Dump the PUFS file directory; assemble and execute the program in files N_1 (version i_1), N_2 (version i_2), etc., using data from the files M_1 (version j_1), M_2 (version j_2), etc. Just *before* execution, the serial and block numbers of the PUFS dump are printed, followed by the command itself. If any of the files cannot be found, the report FNF is printed.

Commands are read from the input tape one at a time and executed. After execution, a copy of the command is printed; this is intended as an indication that, as far as the PUFS program can tell, execution has been correctly performed. EXECUTE is an exception in that the command is necessarily printed before execution.

If a command cannot be recognized, the report E is printed followed by a copy of the command.

Experience with the use of the above commands has suggested that it would be convenient to have a command which enabled the user to re-number a selected section of a file and to re-sort the lines according to the new numbers. This, together with a command for merging two files and re-ordering the lines according to their numbers, would enable new sections to be introduced into the middle of a file, or an existing section to be moved from one place to another, with a minimum of trouble.

Implementation

PUFS was implemented for EDSAC 2 using the WISP list-processing system (Wilkes, 1964). Characters are read by a modified WISP input routine which converts them to an internal 6-digit code, without, however, ignoring spaces after the first or equating commas and carriage returns as is usual in WISP. For the purpose of storage on magnetic tape, characters are packed six to the word, machine-code packing and unpacking routines being provided. When loaded into the main store, lists are stored as ordinary WISP lists with each character occupying the CAR of a separate register. Line numbers are expressed in binary form and also occupy the CAR of a register; when a file is transferred to magnetic tape, however, the line numbers are converted to decimal and stored in character form in WISP code. This is in accordance with the principle that the files as held on magnetic tape are images of what might be printed on a piece of paper.

The number of characters that can be contained in a file is limited by the capacity of the main store, and a large improvement could be made if the WISP system were modified so as to permit the storing, in simple lists, of more than one character per word. However, since the EXECUTE command can take its input from more than one file, no difficulty arises if very long files are split up into a number of sub-files.

Example

The following example shows how the system might be used for keeping lists of members of a society. This example is chosen for its simplicity, and to emphasize that the information is stored in a form quite independent of any meaning it might have, for example, in a programming system. Line numbers are indicated here by having # printed in front of them; in the EDSAC 2 implementation a suffix $_{10}$ is used. Explanatory comments are given on the right.

```
INPUT.  /

MEMBERS
    F. BAKER             Original document
    J. BROWN
    C. A. JONES
    S. ROBINSON
    T. L. SMITH
    A. W. STUART
    S. T. WILLIAMS

#35   K. A. JOHNSON   Correction of errors made in
#70   A. W. STEWART   course of punching
***

PRINT.  /            Copy for checking
FILE.  10 /
INPUT.  /

ASSOCIATE MEMBERS
    A. BIRD              Second original document
    T. A. DICK
    A. S. JOHNSTON
    T. E. YOUNG
***

PRINT.  /                 Copy for checking
FILE.  11 /
LOAD.  10 /
MODIFY.  /

#32   T. JACK             Routine updating of file (two
#65   V. SMITH            insertions and one deletion)
#40
 ***

RENUMBER.  /
PRINT.  /
FILE.  10 /
APPEND.  11 /             Combination of two files
PRINT                     Fair copy (without line
  SUPPRESSED.  /          numbers)
DUMP.  /
*
```

When run on the computer the above PUFS program produces the following output. It will be remembered that PUFS prints a copy of each command *after* any output produced by that command.

```
LOAD PUFS

INPUT.

10   MEMBERS
20      F. BAKER
30      J. BROWN
35      K. A. JOHNSON
40      C. A. JONES
50      S. ROBINSON
60      T. L. SMITH
70      A. W. STEWART
80      S. T. WILLIAMS

PRINT.
FILE.  10
INPUT.

10   ASSOCIATE MEMBERS
20      A. BIRD
30      T. A. DICK
40      A. S. JOHNSTON
50      T. E. YOUNG

PRINT.
FILE.  11
LOAD.  10

32      T. JACK
40
65      V. SMITH

MODIFY.
RENUMBER.

10   MEMBERS
20      F. BAKER
30      J. BROWN
```

```
 40    T. JACK
 50    K. A. JOHNSON
 60    S. ROBINSON
 70    T. L. SMITH
 80    V. SMITH
 90    A. W. STEWART
100    S. T. WILLIAMS

PRINT.
FILE. 10
APPEND. 11

MEMBERS

       F. BAKER
       J. BROWN
       T. JACK
       K. A. JOHNSON
       S. ROBINSON
       T. L. SMITH
       V. SMITH
       A. W. STEWART
       S. T. WILLIAMS
ASSOCIATE MEMBERS
       A. BIRD
       T. A. DICK
       A. S. JOHNSTON
       T. E. YOUNG
PRINT SUPPRESSED.
1301  987  DUMP.
```

Acknowledgements

I would like to express my thanks to Mr. J. D. Blake who made contributions to the later stages of the development of PUFS, and also to Mrs. V. Bayley who assisted with the checking out of the system on the EDSAC. This paper was presented at the International Symposium on Data Processing Machines held in Prague in September 1964. I am grateful to the organizers of the Symposium for allowing it to appear in *The Computer Journal* as well as in the *Proceedings* of the Symposium.

References

Corbató, F. J., *et al.* (1963). *The compatible time-sharing system; a programmer's guide*, M.I.T. Press.

Wilkes, M. V. (1964). "An experiment with a self-compiling compiler for a simple list-processing language," *Annual Review of Automatic Programming*, Vol. 4, Pergamon Press Ltd.

Book Review

Computers and Thought. Edited by Edward A. Feigenbaum and Julian Feldman, 1963; 535 pages. (Maidenhead: *McGraw-Hill Publishing Company Ltd.*, 62s.)

One of the most interesting and yet intractable points of both popular and professional discussions about computers is whether machines can, or conceivably could, "think". Unfortunately it is still necessary to put that last word in quotation marks as we have no agreed definition of the term. Several of the authors represented in this volume attempt that definition, the consensus being an ostensive definition which may be roughly stated as "behaviour indistinguishable from that of a human being under the given circumstances, if that behaviour on the part of a human being would be characterized as thought."

The present volume is a collection of the most notable papers on the study of thought processes by means of highly structured computer programs. The editors label this approach that of "cognitive models" and explicitly exclude work along the lines of self-organizing systems, "neural cybernetics", or homeostatic models. Whatever one's opinion of this or that approach the extent to which the experiments reported here have met with success is indeed the most lasting impression.

The 20 papers report on experiments in artificial intelligence that attempt both to simulate and to emulate human thought processes, and also include some papers that survey the problems and successes of the attempts. Additionally, the editors have included their own commentary to the several sections of the volume, a great help to the lay reader in following the relationships of the various projects described, and have appended an extensive bibliography of some 900 entries that is meticulously indexed under about one hundred descriptors (topic headings). This last was prepared by Marvin Minsky.

Not so long ago it may have been safe to say that research into artificial intelligence, whilst of vast interest, could yield little by way of immediate results of profitable application. Certainly the majority of the papers are concerned with game playing (chess and draughts), theorem proving, and similar intellectual pursuits, but two papers are far more mundane. That by Tonge on line balancing for assembly processes is the more advanced and therefore the more impressive. At the time of his report, first published in 1960, his program was already able to find a practical solution to balancing an actual 70-station assembly line in the appliance industry. If his program were implemented on a current-generation machine (JOHNNIAC was in fact used) it would be faster and probably cheaper than a human solution by a skilled and experienced manager. In the other paper on profitable applications Clarkson reports on a model designed for portfolio selection for investment trusts. At the time of publication this was not yet ready for use. Both papers are included in the volume because they use a heuristic rather than algorithmic approach.

Primarily this is a book for the library or for the specialist. Its value is greatly enhanced by the ample bibliography and by the care taken by the editors to relate their selections to each other and to their place in the field.

H. D. Baecker