To: MTB Distribution

From: Gary C. Dixon

Date: January 13, 1978

Subject: A Tool for Converting Files
Created by IBM PL/I F Compiler
To Multics Format

THE PROBLEM

A conversion tool is need to convert files created on an IBM system to Multics standard format.

Specifically, a tool for converting record-oriented files created by the IBM PL/I F Compiler is needed for the program conversion effort currently underway at the Puerto Rican Highway Authority. The same tool (or a similar tool with only minor modifications) could be used for converting files created by the IBM PL/I Optimizing Compiler. Such a tool would be invaluable in running benchmarks and in future site conversions where an IBM system is involved.

Constraints

The tool should:

1. Process IBM files dumped on IBM standard-labeled or nonlabelled tapes.

2. Be driven by a PL/I structure declaration which defines the records in the file.

3. Run solely under the Multics environment, obtaining only the file itself (and perhaps an include file containing the PL/I structure declaration for the record format) from the IBM system.

4. Convert record-oriented files created by the IBM PL/I F and PL/I Optimizing Compilers, containing only arithmetic, character, pictured and bit string data. The initial implementation of the tools should handle files containing only a single record format. Subsequent implementations might handle files containing self-defining records with variable formats.

5. Convert to Multics sequential or keyed file. If conversion is to a keyed file, the records must contain an embedded key accessed by a character string element of the structure (or perhaps by a structure element that can be converted to a character string according to Multics PL/I conversion rules). The program should NOT require that the records coming from the IBM file be in key-sequential order.

6. The conversion program should generate a summary of the converted file describing: for each PL/I data type, the fraction of the record declared as that data type; the size of each record (in the converted Multics file); the total number of records processed; for keyed files, the lowest and highest keys created; the total size of the records in the file (total bytes converted); plus information about the Multics files returned by vfile_status_.

7. The program should be reasonably straight-forward to use, and should be efficient enough to process files containing several hundred thousand records in a reasonable time. There should be no limit (other than Multics file space limitations) on the total number of records which can be processesd.

8. The program should be written in such a way that a group of files sharing the same record format can all be processed with a minimum of user intervention.

9. The program must be cognizant of all IBM PL/I data typing defaults and structure mapping rules. In particular, the IBM structure mapping rules differ SIGNIFICANTLY from those of Multics.

PROPOSED SOLUTION

The User Interface

A proposed solution which meets these constraints might use the following procedure to convert an IBM data base.

1. The user writes a small PL/I subroutine which contains:

   A. A PL/I structure declaration defining the structure of records in the file. This should be modified to include IBM defaults for data types and alignment. A PL/I default statement could be used for this purpose.

   B. An include file declaring the input argument structure required by a file conversion subroutine (let's call it convert_ibm_file_). This structure should contain: the name of the calling subroutine (for use in error messages and calls to stu_); the attach description defining the input file; the attach description defining the output file; the attach description defining an error disgnostics and summary file; the name of the (major) structure defining the record format; optionally, the name of the structure element containing an embedded key.

   C. Code which initializes the convert_ibm_file_ input structure.

   D. Code which allocates the record format structure (if it is based or controlled) and which references some element of the structure to cause PL/I to generate runtime symbol table entries for all elements of the structure.

   E. A call to the convert_ibm_file_ subroutine, passing as arguments a pointer to the input structure, and a return code.

2. The user then compiles the PL/I program with the -table option.

3. Finally, the user runs the program passing the attach descriptions for input and output files as arguments (or perhaps the user has stored in the input structure to convert_ibm_file_ synonym attachment descriptions which reference I/O switches attached before the program is run).

4. The convert_ibm_file_ subroutine reads records from the input file, and converts them to Multics records which are written into the output file (perhaps using the embedded key from the output record). Conversion continues until the input file is exhausted.

The Conversion Subroutine

The conversion subroutine, as currently envisioned, must:

1. Obtain a pointer to its caller's stack frame for use in calls to stu_.

2. Call stu_$find_runtime_symbol to obtain a pointer to the runtime symbol node for the record format structure.

3. Obtain 2 temporary segments from get_temp_segments_, one for IBM data and one for Multics data.

4. Using the runtime symbol nodes for the record format structure, construct two symbol trees for the structure, one representing the IBM structure and the other representing the Multics structure.

   A. Data from the runtime symbols can be obtained for the Multics symbol tree, including: offset of element from beginning of containing substructure; length of element; element attributes - data type, dimensions, extents, precision and scale, etc; element name; pointers to father, brother, child nodes. Such things as location of picture specifications, and location of structure elements referenced in refer extents, must be handled in this symbol tree.

   B. Similar data must be constructed for the IBM symbol tree, using the information in the runtime symbol nodes plus knowledge of IBM data attributes and structure mapping rules. Equivalent picture specifications and refer extent information from the Multics tree must be mapped into the IBM tree. It is because this mapping of information from Multics to IBM tree must be performed that the information must be stored in the Multics tree originally, rather than using stu_ (the symbol table utility) to extract the information from the runtime symbol nodes as needed. (Also, the cost of extracting the information via stu_ for each record would be exorbitant.)

5. Obtain space at the end of the IBM and Multics symbol trees for the input and output records. In general, the actual size of each record won't be known if it contains refer extents. Space for the IBM records should be aligned on the proper byte boundary, according to the requirements of IBM's structure mapping algorithm. This is strictly not necessary, but will probably provide for best alignment of data to be converted. Note that, under the IBM structure mapping algorithm, a structure containing word- or doubleword-aligned data is not

necessarily aligned on such a boundary as a whole. IBM aligns structure elements, starting with the deepest elements in the structure and working out to the major structure. This leads to highly-packed data, but also leads to different data padding than the Multics structure mapping algorithm.

6. Using the remainder of the temporary segments to hold these records, call iox_$read_record directly to read the records. Then the actual length of read records returned by iox_ can be used to determine how much data there is.

7. Process records from the input file, sequentially until the file is exhausted.

   A. Originally, the tape file should be read using tape_ibm_ with -mode ascii to defer character translation until we know where the character fields in each record are. This will read the tape in 9-mode, storing 8-bit tape frames right-justified in 9-bit Multics bytes. Therefore, arithmetic and bit string data will have to be repacked before conversion.

   B. The input record will be converted, from the top down, on a field by field basis. This will allow self-defining structures (using the refer option) to have the referenced extents filled in before the extents are actually needed.

   C. As the conversion of each record progresses, error diagnostics should be issued for each input field containing invalid input data (ie, data which cannot be converted into the designated Multics output data type). The diagnostics should be output to the same file as the summary statistics (to a file rather than the console since diagnostic information may be lengthy). It should include: record number of record being processed; its key field value, if file is keyed; name of field in error; an octal and EBCDIC (dump_segment_) dump of the field in error; a description of its location in the record. Some default value should be stored in the data field, and the remainder of the record converted. This will facilitate later patching of the field. A count of records in error should be added to the summary statistics. Perhaps the count should be maintained for each record field, as well as for the overall record.

8. Close input and output files; generate summary statistics in the summary file; close this file; free the temporary segments; and return.

IMPLEMENTATION RESOURCES

The implementor of this facility should have had some experience with IBM PL/I data files and data types; should be knowledgeable in the way Multics data types are stored; should have had experience in program debugging interfaces (use of probe, debug, stu_, decode_descriptor_, assign_, pack_picture_, etc); some experience reading IBM tapes.

The time required by such a person to implement, test and document the converter would be about 22 person-days.

1. 3 days for research.

2. 1 day to write convert_ibm_file_ subroutine initialization.

3. 1 day to write symbol tree creation code, including design of symbol tree nodes.

4. 3 days to write and test IBM structure mapping algorithm.

5. 2 days to write/test IBM-to-Multics data conversion routines.

6. 2 days to write declarations, fix typing errors, get program running.

7. 5 days to test program on a variety of data bases (assuming that these data bases are readily available).

8. 5 days to document program usage and internals, submit MCR and installation forms, and cleanup program.

REFERENCES

Manuals

1. IBM PL/I F Compiler Reference Manual (Order No. GC28-8201).

2. IBM PL/I Optimizing Compiler Reference Manual

3. IBM System/360 Principles of Operation (Order No. GA22-6821)

4. MPM Subsystem Writer's Guide, sections on object segment format and stu_ subroutine.

Program Listings

bound_probe_

symbol_attributes.pl1

symbol_name.pl1

bound_pl1_

prepare_symbol_table.pl1

Include Files

std_symbol_header.incl.pl1

pl1_symbol_block.incl.pl1

runtime_symbol.incl.pl1

Other Programs

create_data_segment_.pl1

ebcdic_to_ascii_.alm

assign_.pl1

An IBM to Multics fixed decimal conversion subroutine written by Warren Johnson (at PRHA site).