mxload: Read Multics Backup Tapes on UNIX

W. Olin Sibert

Oxford Systems, Inc.
32 Oldham Road
Arlington, Massachusetts
U. S. A. 02174

ABSTRACT

Oxford Systems' **mxload** package is software for reading Multics backup_dump tapes on UNIX systems. It includes programs for reloading data, for making maps from dump tapes, and for conversion of special Multics data formats, such as archives, mailboxes, and Forum meetings. Wherever possible, Multics attributes (owner, timestamps, etc.) are converted to UNIX equivalents, and numerous options are available for controlling the reload. The product, its style of use, features, limitations, and customer experience are described.

INTRODUCTION

As Multics users integrate other systems into their Multics environments, and migrate Multics applications to those other systems, they are invariably faced with a difficult question: ‘‘How do I get my data over there, anyway?’’

Over the years, various ad-hoc techniques have evolved, but none are particularly satisfactory for large-scale transportation of Multics hierarchies to other systems. While individual files can be transferred fairly easily, it has been extremely difficult to transfer of entire hierarchies while preserving directory structure, names, segment attributes, etc. When migrating to systems without Multics-style hierarchical file systems, such preservation is impractical, but for targets such as UNIX systems, it is feasible and desirable.

This problem is solved by the **mxload** package (pronounced ‘‘em-ex-load’’), which allows transportation of Multics data in `backup_dump` format to UNIX and UNIX-like systems. The package performs automatic conversions of Multics data formats and attributes to corresponding UNIX values. It can operate either in an automatic mode, driven by control files, or interactively, and can reload anything from a single segment from the middle of a tape to an entire hierarchy on a large set of tapes. Because it works from standard Multics `backup_dump` format tapes, it can be used long after the Multics system that wrote the tapes is gone.

EXISTING DATA MIGRATION TECHNIQUES

Before describing **mxload**, it will be helpful to review some existing techniques for migration. They are presented here in rough order from most Multics-specific to least.

`backup_dump` Tapes

Were it not for **mxload**, this method would also be completely Multics-specific. Even so, it is fairly flexible, as it allows data to be copied easily between unrelated Multics systems with complete preservation of attributes and contents. With **mxload**, the `backup_dump` format can be used for arbitrary target systems (but only in one direction—**mxload** cannot create new tapes for transfer back to a Multics system), and, if the tapes already exist, can be used without the assistance of a Multics system. Unprivileged users can create and reload `backup_dump` format tapes, though it is a bit inconvenient.

`tape_archive` Tapes

This is also a Multics-specific mechanism, in that the information about file attributes for files on the tape is kept in special auxiliary files that can be manipulated only with the Multics `tape_archive` command. However, because it uses ANSI format tapes, the stored files can be read by other types of systems, albeit not without considerable manual intervention and name translation. While `tape_archive` preserves the contents and attributes of its files, it does not preserve directory structure, and has no way of dealing with entire directory trees.

ANSI- and IBM-format Tapes

These are tape formats supported by Multics primarily for interchange purposes. They can be created directly by programs using the tape I/O modules, by the `tape_out` command for storage of multiple files, or by the `copy_file` command, for manipulation of individual files. They preserve only the contents of files, not the attributes or directory hierarchy, and therefore require manual intervention when reading to determine what to do with the data. When moving data to UNIX systems, these formats (like `tape_archive`) also have the disadvantage that most UNIX systems have what could charitably be called primitive tape-handling software. Often, the only form of tape that can be read reliably on a UNIX system is unlabeled, not ANSI or IBM labeled formats.

File Transfer Protocol (FTP)
: This is a meaningful option only for Multics systems connected to the Internet (ARPAnet / MILNET). It is suitable only for transferring file contents, not attributes or directory structure, and can transfer files only to other systems on the Internet. Unlike the tape transfer methods, this has limited (but respectable) bandwidth.

Serial File Transfer Protocols (Kermit, xmodem)
: These are programs which can be used to transfer files (contents, not attributes) between Multics systems and any other systems implementing the protocols. Fortunately, the protocols are widely implemented, particularly on small computers, so these can be very effective for small-scale transfers. The bandwidth is limited by communication lines, however, so they are not feasible for large transfers.

`dial_out` Transfer
: This is another technique for transferring data over serial communication links. Unlike the file transfer protocols, however, it has no error checking and no handling for file names. In general, it is suitable only for very small transfers that will be inspected before use.

Existing Multics file transfer techniques are seen, then, to fall into two[1] categories: one characterized by transferring individual files, requiring manual intervention for each file, losing file attributes, etc., and another characterized as being entirely Multics-specific, and not useful (without **mxload**) for transferring data from a Multics system to something else. There was no standard mechanism for taking a Multics directory tree and re-creating it, as accurately as possible, on a non-Multics system.

SOLVING THE DATA MIGRATION PROBLEM

The **mxload** package was developed to address the inadequacies of existing file transfer techniques. The goal is simply stated:

> Provide a program for transferring, with a minimum of manual intervention, the contents and attributes of Multics segments and directory trees to UNIX systems.

UNIX systems were chosen as the target because they seemed to be the only widely-used systems with a Multics-like hierarchical file system. In practice, **mxload** can operate on other systems as well: it was developed in large part by reloading Multics data onto MS-DOS diskettes, and one

[1] Actually, there is also a third category, in that there exist private implementations of Multics commands for handling the UNIX standard **tar** and **cpio** tape formats. However, these are not generally available, and have limited capabilities for attribute preservation. Also, they do not provide any way to handle existing Multics backup tapes.

mxload user has successfully modified it to run under IBM's VM/CMS system. In principle, it would be easy to convert to any system with a hierarchical file system, such as Data General's AOS/VS, Prime's Primos, or Stratus's VOS. More conventionally, **mxload** runs on most UNIX-derived systems, both the System V (release 2 and release 3) and Berkeley (4.2bsd and 4.3bsd) variants. It has been run successfully on DEC's VAX Ultrix systems, on Sun's SunOS (releases 3.5 and 4.0) systems, and, of course, on Bull's XPS-100 systems.

Satisfying the need to run on many different target systems made it necessary to distribute **mxload** as source code. The potential user community has many different target systems, and distribution of that many binary versions would be impractical. Source distribution makes it easy to modify **mxload** to accommodate the quirks of different target environments. This also makes it easy to use **mxload**'s internal routines in writing additional specialized conversion programs.

The Multics `backup_dump` tape format was likewise the obvious interchange medium. It would have made no sense to invent a new one, since `backup_dump` format already had all the necessary information. It was also a goal to be able to read existing `backup_dump` tapes; that is, tapes which people may have already, even though they no longer have a Multics system (this was the author's own problem!). Neither of the other two Multics dump format tapes were as appropriate, since the goal is to transfer directory hierarchies, rather than entire physical disk volumes.

It didn't seem important for **mxload** to be able to create new tapes on the UNIX system, at least not for Multics to read. UNIX systems already have some satisfactory tape backup software (**tar** and **cpio**), and there simply wasn't much demand for sending things back the other way.

Operating Modes

After some research, it was clear that there are two fundamentally different ways that people want to approach the data migration problem. One way is the ‘‘batch mode’’: make a bunch of tapes on the Multics system, carry them to the UNIX system, reload them all at once, and end up with a UNIX system with user directories pretty much identical to what they were on Multics. The other way is the ‘‘interactive mode’’: make a tape of some Multics directory, carry it to a UNIX system, reload some files from it, and shuffle them around to their desired arrangement.

Initially, it seemed that the batch mode would predominate. This drove the design of the **mxload** control file, which can specify arbitrary translations between Multics and UNIX pathnames, translations from Multics Person and Project IDs to UNIX User and Group IDs, options for handling Multics ACLs. In principle, one could design a set of control files for copying the entire contents (or at least the user directories) of a Multics system to a UNIX system (or systems), translating all pathnames, ACLs, and ownership according to a pre-defined migration plan. In practice, this doesn't seem to work out—perhaps only some Multics users are migrating, or not all Multics data needs to be (or should be) copied, or the migration happens over a long period, a few users at a time, or something else makes automatic operation impractical. Invariably, manual pruning and reorganizing must be done once the data has been migrated. Control files help a great deal, but cannot do the whole job.

The interactive mode, on the other hand, is just that. A user with a private backup tape, for instance, can interactively list the tape's contents, pick individual segments or directories to reload, and perform further conversions on reloaded data, without ever needing to know about control files. This same operating mode is easily used for selective retrievals from a set of system complete dump tapes. The defaults are arranged to facilitate this mode as much as possible.

Data Translation

The other major technical problem is how to get Multics data (including segment attributes) into a useful form on a UNIX system; this clearly requires considerable translation. The basic problem is that Multics uses 9-bit characters, packed four in a 36-bit word, whereas most UNIX systems use 8-bit characters. Sometimes there is a direct equivalence between the 9-bit and 8-bit characters, but other times special handling is required. Some Multics data can't be effectively translated, of course; for instance, by default, **mxload** discards Multics object segments, as they are unlikely to be of any use elsewhere.

In the simple case where a Multics segment is all ASCII (more technically, all the characters in the segment have values in the range from 0 to 377 octal), the translation takes each Multics character and copies it into a UNIX character. If a Multics segment contains data that doesn't fit in 8-bit characters, and isn't recognized as one of the special types described below, it is not translated, but copied bit-by-bit, with each group of eight 9-bit Multics characters copied into nine 8-bit UNIX characters.

More complex translations are possible for Multics archive segments, Multics mailboxes, and Multics Forum meetings. These translations can either be performed automatically during a reload, or manually afterward by stand-alone commands that are part of the **mxload** package. By default, for instance, Multics archive segments with all ASCII components are automatically converted into a UNIX directory containing one UNIX file for each component. Multics mailboxes and Forum meetings can likewise be converted (though this is not done automatically by default), either into a single file containing the text of all the messages (or transactions), or into a directory of individual files, each containing a single message (or transaction). The stand-alone utility programs **mxarc**, **mxmbx**, and **mxforum** can be used to perform these conversions after data has been reloaded.

Many other special Multics data formats are simply left alone by **mxload**, because no automatic conversion seemed feasible or useful. This includes queue message segments, keyed `vfile_` files, other forms of multi-segment file, stored binary floating point data or PL/I structures, and so forth. With the subroutines provided in the **mxload** package, however, it is possible to construct special-purpose translation programs for almost any special Multics data format.

For segment attributes, an attempt was made to figure out how they would best map from Multics semantics to UNIX semantics. This involved many difficult choices, and the result is still not wholly satisfactory. In any case, if **mxload**'s defaults and options are insufficient, the dump map it produces listing all attributes in original Multics form can be used to make corrections. One of the biggest problems is name translation. Some UNIX systems support only 14 characters for file names, whereas Multics supports 32. On such systems, the problem is handled by detecting duplications and constructing names with numeric suffixes, which is indicated in the map. The **mxload** User's Manual describes all the rules in detail.

USING THE PACKAGE

There are six programs in the **mxload** package. The most important is **mxload** itself. It is used for all reloading operations: it takes instructions from control files (if any), reads data from a `backup_dump` format tape (or equivalent), produces a reload map, and creates files containing the reloaded data. While reloading, it can perform data conversions automatically; by default, ASCII files are converted, ASCII-only archives are unpacked into individual component files, object segments are discarded, and all other data forms are reloaded as binary data, for later processing by a stand-alone utility.

Another of the programs is **mxmap**. This produces a map of a tape's contents, without reloading any data. It is useful for testing and for checking whether particular data is on a tape. Because it only reads data without reloading, it is much faster than **mxload**.

Finally, there are four stand-alone utilities: **mxarc**, **mxmbx**, **mxforum**, and **mxascii**. These are used to convert special Multics data formats (archive segments, mailbox segments, Forum meeting directories, and arbitrary binary data, respectively) to useful UNIX equivalents. Archive components can be extracted individually or all at once by **mxarc**, and for convenience, an archive's table of contents can be listed without extracting anything. Mail messages and Forum transactions are extracted by **mxmbx** and **mxforum** and can be placed either in individual files or in a single file containing all the messages or transactions concatenated, separated by formfeeds. Arbitrary binary data can be converted from 9-bit to 8-bit form with **mxascii**, which trims off the high-order bit from each character.

There are several ways to provide **mxload** with Multics data to reload. Writing a `backup_dump` format tape is most straightforward, but **mxload** is by no means limited to physical tapes.

Reading a `backup_dump` format tape may prove difficult or slow on some target UNIX systems: many UNIX systems have trouble with tape in general, and often their tape drives are very slow—a far cry from the typical 200 inch per second Multics tape drive. Consequently, it may be faster to use a copying utility (like **dd**) to copy the tape's entire contents into a UNIX disk file, and then process that file with **mxload**.

Sometimes, for various reasons, a `backup_dump` format tape may not be readable at all on a UNIX system. In this case, it is best to create a `backup_dump` file on Multics, using the Multics `backup_preattach` command to create a Multics multi-segment stream `vfile_` containing the image of that would be on the tape. This file can then be copied, as a single unit, onto an unlabeled tape and transferred to the UNIX system. Once there, it can be processed by **mxload**.

The `backup_preattach` technique can also be used when tape is not available, or for individuals to use in transferring small hierarchies. The Multics file it generates can be transferred to a UNIX system by one of the file transfer protocols and then processed by **mxload**. This can even be done by transferring the backup file to a diskette.

Reloading with **mxload** is fairly expensive in processor time because of the 9-bit to 8-bit translations, and this is often slow enough that the tape drive cannot be run at full speed, introducing additional start/stop delays. In some cases, it may be preferable to do large reloads from disk images of `backup_dump` tapes, rather than from the tapes themselves.

PRODUCT EXPERIENCE

Although there are several operating **mxload** users, only one has made extensive use of as of this writing. Their experience with the UNIX version (on a Gould NP-1 system) was very successful: they were able to bring up the beta-test release **mxload** and use it in a day or so, and have since reloaded megabytes of Multics data with it.

The bulk of their experience, however, is with the VM/CMS version that they created themselves, and it doesn't bear directly on using **mxload** to migrate Multics data to UNIX systems. Once all the difficulties of the VM/CMS environment were overcome, **mxload** was used to reload several gigabytes (40 tapes) of Multics data as VM/CMS files, which were then distributed semi-automatically to their owners through an extra program written for the purpose. Because of difficulties with VM/CMS, all the tapes were manually copied to disk files and then processed by **mxload**.

Other users' experience on UNIX systems has been positive, though less extensive. Bringing up **mxload** has been straightforward every time. In general, the only porting difficulties have involved differently named include files or occasional missing C-library functions. For this reason, **mxload** includes versions of any library functions which have been unavailable in different target environments.

Because there has been relatively little UNIX experience, and the VM/CMS environment does not support those features, relatively little use has been made of **mxload**'s name and attribute translation features. It remains unclear how these will fit into a well-planned migration effort.

SUMMARY

The Multics data migration problem has been described, as have various existing techniques for addressing it. The difficulties of those techniques were explored, and an explanation of how **mxload**'s goals address those difficulties was given. Finally, the features of the **mxload** package were described briefly, along with a summary of experience with the product.

This paper's description of **mxload**'s operation and capabilities is necessarily limited. A complete description is provided by the **mxload** User's Manual, available from Oxford Systems, Inc.