To: Distribution

From: David M. Jordan

Date: January 26, 1977

Subject: Batch Processing on Multics

I. Introduction

In the past, Multics has led the relatively pristine life of a pure time sharing system. With a small user community, pressures for support of the more mundane forms of computing services were limited and, to a large degree, ignorable. Recently, however, several user sites have been added for organizations whose users are neither sophisticated nor dedicated. In many cases these users are unimpressed by those things that Multics does well, as they are unable to accept the difficulty of performing tasks traditionally associated with large scale systems.

In particular, many users are requesting that Multics provide appropriate mechanisms for the handling of punched cards in what can best be described as the "traditional" manner. These users are asking that Multics not only read cards, but that it also allow the cards to be processed as batch input. From the users' viewpoint, this implies such things as the ability to read source, object, and data in a single deck and includes the ability to control the job through some sort of batch monitor which implements a fairly traditional form of Job Control.

II. Major Areas of Concern

In order to provide a reasonably broad implementation of batch capabilities some areas of the system will need to be reworked and some completely new code will have to be generated. In each instance, the object of the changes is to provide the end user with a reasonably powerful, consistant, and easy to use form of batch processing.

A. Security

The concept of security is a basic element of the Multics design, and every effort should be made to provide as much security as is reasonable in a batch environment. However, it

---

Multics Project internal working documentation. Not to be reproduced or distributed outside the Multics Project.

seems clear that the whole concept of batch processing limits our ability to provide real security: a card deck may be easily "borrowed"; identification of a remote batch station may not be possible with any real assurance. In spite of these problems and others which may become evident later, the needs of the users must be considered primary, and thus we must be careful to provide a usable facility.

One major concept to be considered is an individual site's ability to determine, for itself, the level of risk involved in providing batch services. In fact, the site should be required to take positive action to enable any batch processing at all and further, to allow those services which are more risky.

B. Remote Batch Processing

The use of remote batch workstations is becoming more widespread as time passes. In fact, some Multics users have literally hundreds of stations which they would like to interface to the system. Although we have provided some remote support in the past, several changes appear to be required in order to support batch processing in a consistant manner:

1. Initialization of Remote Connections

   Problems currently exist in terms of initiating a dialup remote station through the Initializer. Most of these problems are probably due to bugs in the current software, but, when we begin to consider "hundreds of stations", this whole area probably requires reexamination. This is especially true considering the desire to allow the submission of jobs, not just decks, from the remote station.

2. Batch Processing Input

   Currently Multics allows the use of a remote station to read card decks into the card pool, but not to submit them as batch jobs.

3. Other Considerations

   Several other factors are of interest relative to remote stations: the structure of a remote deck should be as identical to a locally submitted deck as possible; the source of the remote input should be available to the running job to allow automatic routing of output; the ability of the IO daemons to handle hundreds of queues should be examined.

C. Local Batch Processing

The major consideration relative to local batch processing is that its use be essentially identical to the use of remote batch.

D. Card Input, Batch and non-Batch

1. In order to provide as much consistancy as possible, the card deck structures, for both card input (as currently defined) and batch input, should be essentially identical. The only differences should center around whether the deck is to be executed or not. Also, the procedure which reads cards should be able to read both card input and batch input decks without any operational intervention being required.

2. The current requirement that a card deck be in a single card format should be abandoned or significantly modified. It is not unreasonable to assume that users of a batch facility will want to mix source, object, data and control cards in a single "job". In order to make this feasible, it will probably be necessary to read the entire deck (with the possible exception of overall deck control cards) in raw mode and to leave the actual format conversions to either the copy_cards command or to the batch processing control procedure.

3. In order to allow reasonably natural use of cards, a new punched card format should be defined which, like the "MAP" preaccess command, will cause automatic conversion of upper case to lower case with appropriate escapes to get upper case input.

E. Batch Processing Environment

It seems both reasonable and proper to allow a card deck to be executed as though it were simply an absin segment. In this mode of execution, however, the user is restricted by having to anticipate what the system will want next. The Multics command language has been designed primarily as a conversational interface with the user prompting the system and the system prompting the user. In a batch environment, however, this can be extremely clumsy. If the user executes a Fortran program involving I/O, for example, he must remember to include a mechanism for answering the "Close files?" question. In addition, it can be extremely difficult to handle errors in execution, as abs_io_ will generally refuse to continue, thus, as a result of an error, no output at all may be printed.

In order to allow more reasonable use of Multics in a batch mode, it therefore seems necessary to provide some sort of batch job control procedure. Such a control procedure would provide features associated with batch such as a job control language, error handling utilities, i/o control functions and so on. All in all, the design and implementation of such a batch control system may be the most significant effort required to produce a useful native Multics batch system.

III. Summary

The preceding paragraphs have outlined some possible design objectives of an easily usable batch facility for Multics. It seems clear that more detailed design work will be required and that a significant effort will be needed to implement such designs. However, the Multics environment provides many of the tools necessary for the implementation and thus the overall effort should be quite reasonable.

While there can be no doubt that a batch system as described above will violate some basic Multics design philosophies, we must remember that it will also open the system for a wide variety of less technically oriented end users and will therefore add significantly to the utility of the system.