To: Distribution

From: Bernard S. Greenberg

Subject: On the Cardreaderless Unified Bootload Tape

Date: 10/18/74

On the Cardreaderless Unified Bootload Tape

I. Introduction

It has been noted for some time now that the successful bootloading of Multics is dependent upon the proper functioning of the installation card reader, which is rarely used at any other time in Multics or BOS operation. Furthermore, it has been noted that Multics is dependent upon an offline T&D program to load Microprogrammed Peripheral Controller (MPC) firmware. What is more, the amount of tape handling required to bring up Multics, or even bring it down, seems excessive. Dialogue is required with these three different operating systems, and offline T&D. This MTB proposes a single bootload tape strategy which addresses all of these problems, and attempts to put the bootload tape house in order.

II. "One Multics, One Tape"

The multiplicity of required tapes is more than a tape-handling problem. The issue of synchronizing compatible release of Multics, BOS, and the Salvager has created much confusion due to hardware and software changes, e.g., new directory formats, the 6100 cache, etc. Much interest has been expressed in a scheme whereby all of these subsystems, which are, in truth, parts of Multics, the Multics hard core being only one of these parts, appear on one tape. The distribution of software releases *in toto* on a single tape is attractive from operational and marketing viewpoints, as well as allowing ready identification of software cited in trouble reports.

Hence, we propose a single bootload tape, which will contain BOS, Multics, a Salvager, a Firmware loader, possibly selected hierarchy segments appropriate to a Multics release, and all data objects required by these programs that do not reside in the Multics hierarchy.

This tape would be a self-loading, i.e., hardware bootable tape. This implies that it is not a Multics Standard Tape. It is folly to believe that all tapes handled in a Multics installation are Multics Standard Tapes. The author has personally dealt with many non-Multics tapes on Multics, and has had to suffer because of the lack of attention given to nonstandard tape software because of this misbelief. In fact, the insistence on BOS being a standard tape has led to the card reader necessity, which we will discuss further below.

The format of the tape would be as follows: the tape contains a series of objects, each preceded by a header, which describes it. Each header begins on a new physical record. Each object begins on a new physical record, and is generally of fixed block size. The length, in words, of each object, is specified in the header. The version, date of generation, type of object, and other descriptive information are provided in the header.

Several objects are special. The program bootloaded by the hardware clearly can have no header. It must dynamically construct its own header for later reference, and be otherwise special-cased. Virtual-memory operating systems, specifically, Multics and the Salvager, follow their headers as other objects, perhaps with an intervening end-of-file mark for ease in searching; but other than that, they appear on the tape as they do today. They are followed by a double file mark so that they may be scanned over easily.

BOS is the principal domain of the tape. BOS programs are tape objects, as are Runcoms, DATANET images, etc. BOS is responsible for the manipulation of the tape and its contents. BOS will maintain (in core and on BOS disk) a directory of all objects on the tape, up to the first virtual memory operating system. Some of the objects in this directory will have copies on BOS disk, as indicated by appropriate flags in the directory.

The current types of objects being contemplated are:

1. Bootload program
2. BOS programs
3. DATANET images
4. Firmware Files
5. Runcoms
6. CONFIG Decks
7. Virtual Memory Operating Systems
8. Reload tape images

Some of these will be discussed separately later on.

III. Firmware, Zero-Six-Dog, and HFED

The Microprogrammed Peripheral Controller (MPC) required by Unit Record Peripheral equipment, MTS500 tape subsystems and DSS190/191 Mass Storage Subsystems require special programs known as 'firmware' to be loaded into them at the time they are powered up. Normally, these programs stay running as long as the MPC is powered up. However, on certain rare occasions, and on powering up the system, it is necessary to load this firmware. GCOS, the other series 6000 operating system, has programs for doing this. Multics has none. Thus, Multics relies on an offline T&D (test and diagnostic) program, PRG06D (known affectionately as 'zero-six-dog', in field engineering jargon) to load firmware. This program runs under the T&D "PAS6000" executive, and those who must load firmware at Multics sites must either be field engineers, or otherwise become conversant with "06D" and the PAS6000 executive. These programs have nothing in common with Multics or its subsystems, and their dialogues are lengthy and arcane.

To deal with this need, the author has developed a self-loading tape which loads firmware for Multics, holding a minimal conversation, following BOS conventions. This tape has all the necessary firmware programs on it, and is very easy to use. This tape has been in regular use by CISL Development Operations for some months now. While this development has been met with much enthusiasm from those quarters, some doubts have been raised by HFED (Honeywell Field Engineering Division) because of the fact that this tape has firmware programs on it.

HFED maintains a practice of maintaining "firmware tapes", which contain all 6000 MPC firmware in a consistent released state, carefully coordinated with well-documented hardware change notices (FCOs) for the MPCs. Current practice maintains that these tapes are to be 'the only source of firmware' at a site. This is so that these tapes can be withdrawn and replaced with

later revisions when the latter are released, and thus resolve all ambiguities and questions about the revision level of firmware being used at a site. In order for this to work, 6000 System software must not "embed" copies of firmware on tapes, disks, etc. In fact, GCOS does precisely that, copying the firmware tape onto disk.

The requirement of mounting a separate firmware tape (always on a separate drive than the T&D program tape) is one of the highly undesirable features of PRG06D. In fact, the use of the unloaded tape controller requires the operator or field engineer to throw console switches on the MPC to inform it of what drive will be used for the firmware tape in the middle of the conversation with PRG06D.

Thus, we propose that we embed firmware programs on the 'unified Multics Tape', with a reasonably multicious program for loading them. This program, probably about identical to the one currently in use at CISL, would give identifying information about firmware versions to resolve ambiguity. Clearly, when new firmware tapes are issued, a new version of the unified tape must be created (see "Generation" below). The Multics firmware loader program should have the ability to read the standard firmware tape, and copy files from it onto BOS disk.

Another advantage of keeping copies on disk is the ability to load firmware and run ITRs and MDRs (test firmware) under online T&D. The new I/O interfacer has been designed with this capability in mind.

IV. Card Readers and CONFIG Decks

Other than use by the I/O Daemon, the card reader at a Multics site is used only to bootload from the hardware, read a card telling BOS where on disk to put itself, and load the CONFIG deck. It seems eminently reasonable, considering the small extent to which cards are used in Multics, that one should be able to accomplish these functions without a card reader, and thus, the incapacitation, or even nonexistence, of a card reader would only be a minor loss.

The reason that we cannot bootload BOS from the hardware is that current BOS insists on being a Multics Standard Tape. We have already attacked this problem.

The questions of where BOS ought put itself on disk and the specification of the CONFIG deck are related. We propose to have named CONFIG decks on the unified tape. A unified tape, as received at a user site from Honeywell, would contain no such

decks. Via the customization procedures specified under "Generation" below, such decks are added to a customized tape, and named.

One CONFIG deck is active and valid at a time. The command "CONFIG STORE FOO" would cause the active deck to be saved on BOS disk as a CONFIG deck named FOO. This is usually useful for later writing out to a customized tape. The command CONFIG L FOO would cause FOO to be read into BOS Common as the active CONFIG deck. All other BOS CONFIG commands work as today, on the active CONFIG deck.

The CONFIG deck can always be created by hand, via the operator's console, if there is no deck on the tape already loaded from cards as presently, or simply ignored for those BOS commands for which it is not needed. The question of the so-called "COLD DISK" card, which tells BOS where to place itself will be considered in the next section.

V. Diskless BOS Operation

Many BOS commands, such as TAPED and PRINT do not need a disk to operate. Others, such as Dave Kayden's disk formatter, may not want to assume a disk to operate. (This is to say that one may wish to format the BOS disk.) The firmware loader clearly assumes no operational disk when it runs.

Hence, we have chosen to generalize BOS to diskless operation. CONFIG decks can be loaded, edited, and generated with no disks present. Once a PART BOS card and an appropriate D191 (or whatever) card have been defined, the command "LOADSK" may be issued to cause BOS to scan the 'unified tape', loading objects onto disk. Until that time, BOS works without use of a disk: modules are loaded from tape as needed.

The BOS listener ('SETUP'), the disk formatter, the CONFIG deck editor (CONFIG), and the firmware loader can all operate easily in this environment.

Until the LOADSK command is issued, there is no directory. Each need for an object on the tape causes a scan ahead until either the object is found, or an end of file (first virtual memory operating system). At this point, the tape would be rewound and searched until it was where it had been. The current CISL firmware loader works this way, which optimizes tape motion.

The LOADSK command causes the tape to be rewound, and all of the objects between the start of the tape and Multics scanned. Those marked in the header as "load-me" are written to BOS disk: the cards defining the BOS partition and its disk must be in the

CONFIG deck this time. Whether an object is loaded to disk or not, a BOS directory entry is still made. Thus, modules may STILL be loaded from tape, even after the LOADSK command has been issued.

Furthermore, objects may be loaded for execution or loaded to disk from tapes on other drives, including special tapes containing only one or two objects. It is even reasonable to load segments off 7-punch cards.

VI. Runcoms and Default Actions

Runcoms are a type of object on the unified tape. Once the old Runcom editor is reinstalled (it has recently been removed) one can edit Runcoms. The current facilities to create and destroy Runcoms are adequate.

One envisions the unified tape being hardware bootloaded, and immediately loading MTS500 firmware automatically. GCOS uses a technique where tape firmware with some known validity is loaded as part of the startup deck, and later versions of firmware may be loaded after this. This is a reasonable policy, as it allows full use of tape for the whole startup operation. (One can read tape forward, but not rewind or backspace without firmware.)

Upon coming to command level after automatically loading tape firmware, one might say 'CONFIG L STD', loading an already-saved CONFIG deck describing perhaps one of several standard configurations at a site. One might then make configuration changes, if one desired, and then type "MULTGO" or whatever, invoking a Runcom of that name which might look as follows:

```
FWLOAD D191
FWLOAD URMPC 3 PR3 CRZ CPZ
TEST PART BOS WRITE
LOADSK
LD355
BOOT
SALV
```

Note that RUNCOM must be capable of operating at most one level deep entirely in core: a buffer in SETUP will be required.

One could even customize a series of Runcoms with CONFIG L commands within them, reducing the bootloading ff Multics to the typing of a single line.

VII. Generation

The unified tape, as distributed, will be generated, by means of a header file and segments in on-line libraries by a new program similar to the MST generator. The images of virtual memory operating systems will be copied from tapes generated in the current fashion, and firmware obtained from HFED firmware tapes. This will require changes to the standard and nonstandard tape dims to support a mode of attachment and detachment which leaves the actual tape attached to the process and not rewound. Thus, Multics standard and non-standard formats may be intermixed on the same physical tape. This allows the current Multics initialization software to be used unmodified. Perhaps it ought know not to rewind the tape if BOS passes it a flag not to do so, allowing the tape to remain positioned in front of the Salvager.

A tape, when received by a site, contains no Runcoms or CONFIG decks. A site booting such a tape must load their CONFIG deck via the console or from cards. This CONFIG deck is necessary to load other firmware than tape, or perform a LOADSK. At this point, the CONFIG deck may be saved to disk, and other CONFIG decks and RUNCOMs loaded, perhaps from cards, to disk.

At this point, a program, perhaps known as EDITAP, would be invoked. It will prepare a file which is essentially a map of the current BOS directory, sorted to place all CONFIG decks before all firmware, which precedes all else. The order of modules on the original tape will not be changed. One may edit this map, deleting objects, and adding names of objects to be loaded from (other) tapes and/or cards. Firmware tapes may also be read here.

A new unified tape will then be written, with the modules specified, and the virtual memory operating systems of the original tape. Thus, a customized tape capable of generating further tapes is produced. Note that any tape may be loaded in any hardware environment, as long as adequate firmware exists on the tape. The CONFIG deck information on a tape in no way limits the generality of a tape, nor restricts its ability to self-duplicate.

The issue of whether or not the replacement of the virtual memory operating systems on a tape should be allowed is an open question: it involves integrity issues, but is probably a good idea.

VIII. Booting and Segments

The "Multics System Tape" exists as a special object on the unified tape. Other than not starting at the beginning of a

reel, it is a Multics Standard Tape in appearance. The first record will be read by BOS as today, and it will load itself as today. The Salvager operates similarly. Both systems are found by BOS via tape scans, finding 'operating system' type objects. The Salvager should follow Multics, to allow running it after Multics without rewinding. This requires some cleverness and heuristics by BOS to validate the position of the tape after running either.

It seems reasonable to include objects which look like portions of reload tapes, to load hierarchy segments associated with a particular Multics release, e.g., a metering program changed because of a hardcore change. Such tape fragments could be reloaded by Multics via the proposed tape dim attach and detach calls.

Another strategy which has been suggested here is to move all non-hardcore segments associated with a particular release into Collection 3 of the MST, perhaps making it quite large. On a cold boot, Collection 3 would be loaded in its entirety. On a warm boot, it would not be loaded at all. This idea has some historical precedent. In other cases, it would be loaded explicitly by Multics command. In this case, BOS will tell Multics it is being loaded on a unified tape.

IX. Acknowledgements

The author wishes to thank Noel Morris, Steve Webber and Dave Vinograd for helping this idea along and suggesting much of its current form.