To: Distribution

From: T. H. Van Vleck

Date: 01/13/75

Subject: Unattended Operation of Multics

Several Multics installations have been attempting to run their systems without an operator present. Although this is not an advertised feature of Multics, these sites have done pretty well by leaving the system running when the operators go off shift, with the understanding that the system will be available without backup or tape mounting facilities, and that if the system crashes it will stay down until an operator comes in. This memorandum describes changes to the supervisor and support procedures which would make unattended operation easier and more reliable. With some very simple changes, unattended mode becomes usable almost immediately; later changes which make additional improvements in unattended mode are also described.

AUTOMATIC RESTART AFTER A SYSTEM CRASH

When Multics crashes, the operator usually brings the system back up again. Almost all of the steps which the operator takes can be automated. These steps are:

1. Determine that the system has crashed.
2. Invoke the "CRASH" runcom.
3. Invoke the LD355 and BOOT commands.
4. Reply "startup" to the Initializer process.
5. Start the Backup and IO daemons.

Existing facilities are sufficient for automating many of these steps. Only minor changes need be made to fix the others.

Determining When the System Has Crashed

Crashes will be detected when the system returns to BOS from Multics operation. There are some cases of system crash such as power failure, idle loop, or hung Initializer process which do not return to BOS: these cases are not handled by the new

scheme, but are fortunately fairly rare. There are also cases in which the system returns to BOS but has not crashed, for instance after a normal shutdown, or in response to the system_control_ "bos" command; modifications will be made to the supervisor and to BOS to detect these cases.

We will set up one word of storage in the BOS toehold for intercommunication between Multics and BOS. This word will be used as a set of 36 switches which will be set either by a BOS command or by a Multics privileged call. One of these switches will be reserved to mean "automatic reboot mode is on"; this switch will be turned ON by a Multics command, and will be turned OFF by Multics command, BOS command, or by automatic crash-loop detecting code in Multics startup. Another switch will mean "the system crashed": this switch will be set ON at boot time and reset by normal shutdown and calls to pmut$bos_and_return.

When the system returns to BOS, any runcom which contained the BOOT command will continue on to its next statement. This statement will be a test of the "crashed" switch in the standard runcom, to discover whether the system crashed or shut down normally.

It is already possible to distinguish between the case of a Return-To-BOS caused by the Multics software and one caused by an XED 4000 from the processor panel. Since the latter case implies that someone is present in the computer room, it should not be assumed that this action is a "crash." Similarly, the setting of the "crashed" switch will allow BOS to distinguish between calls to pmut$bos and pmut$bos_and_return; the second call is made only in response to manual intervention and so should not be detected as a crash.

Some changes could be made in the scheduler to detect idle loops, and it should be possible to set up a detector for the cases in which the Initializer process is hung: but these are refinements which can be proposed separately. A special command to cause the system to crash should also be constructed which can be used by trustworthy installation personnel in cases where they notice that the system should be restarted. (Special access privilege will be provided on the gate which performs this function.)

Invoking the CRASH Runcom

When the operator discovers a system crash, he usually types CRASH immediately, unless it is obvious that a hardware failure will prevent recovery from working. The usual recovery procedures consist of the following commands:

```
FDUMP (unless inhibited)
FD355
```

```
BLAST CRASH
ESD
SALV (if necessary)
LD355
BOOT
```

The actual runcoms will of course be much more complicated, in the style of MOSN-241. Various switches will be added to allow the operator to request that the system pause before booting and salvaging, after crashing, and so on. Similarly, modes to suppress crash dumps altogether or to take printer and/or tape dumps in addition to the FDUMP will be defined.

In order to prevent the system from cycling in a tight loop of boot - crash immediately - recover - boot, several small modifications will be made to the scheme so far outlined, so that the default after an automatic reboot is to turn off automatic mode, until the answering service determines that the system is truly up, and is not in a crash loop. The advantage to this approach is that the full Multics environment is available for programs which try to detect the loop. Automatic rebooting will be discontinued after N crashes after automatic mode was entered. It will also be turned off if there are more than M crashes in K minutes; and admin commands will be added to the Initializer to allow the setting of these parameters.

Invoking LD355 and BOOT

The restarting of the system will be done as a part of the runcom loop which the system will be trapped in, unless a "manual intervention" switch is set or automatic mode has been turned off. Tape-positioning operations may have to be inserted in the runcom to position the unified bootload tape (see MTB-130). The drive on which the unified tape is mounted must be unavailable for use by the supervisor tape-assigment code, so that the tape can remain permanently mounted.

Replying to Initializer

When the Initializer has completed hardcore initialization, it enters the ring-1 environment and waits for an input command from the master console which tells it whether to do a cold boot, enter BOS, or start up the system. The system_startup_ program must be modified to check the switches in the toehold and to manufacture a "startup" command if "automatic reboot" mode is ON. The ring-1 environment will then proceed to call system_control_$startup for answering service initialization.

Starting the Daemons

Most installations have their system_start_up.ec arranged so that the daemons do not automatically start running, but only come to command level, so that operators may modify parameters or choose not to run the daemons at all. But when no operator is present, the daemons will then hang. A simple solution to this problem is to have running unattended imply running without any backup or IO daemons. Alternatively, the standard action taken by system_start_up.ec could be to include the startup of the daemons. Neither of these choices is attractive. The best solution is to make an active function available which will return "true" when the system is in "automatic reboot" mode, so that the system_start_up.ec can take a default action when the system is unattended, and otherwise wait for the operator.

FACILITIES AVAILABLE IN UNATTENDED MODE

The other duties of the system operator relate to the tending of the printer and the mounting of tapes. If we provide a per-system switch which tells whether the system is unattended, it will be fairly easy to cause the system software to reject all requests which cannot be handled automatically, and to suppress messages which have no function except to prompt the operator. For example, user tape-mount requests which cannot be satisfied without human intervention should be rejected by the tape-management software. Similarly, there seems to be little harm in attempting to run the IO daemon, but when the printer paper jams or runs out, the message describing this situation need be printed only once; the device driver should then wait quietly for someone to fix its problem.

Once "unattended mode" is set, user programs should be able to determine that there is no operator present, and when one will be available, if this is known. A flag in whotab will be used to indicate the system mode, and a few more words to show the time this mode will change. The system mode and schedule data will be set by issuing an Initializer process command. (Perhaps adding and deleting an operator should be thought of as dynamic reconfiguration.) When the system changes from unattended to attended, the flag in whotab may be used, at installation option, for some other information such as the initials of the chief operator on duty. The subroutine system_info_ and the command/active function "system" will be updated to return these new items.

BACKUP

The major problem with running the system in unattended mode is that the incremental backup daemon will eventually fill its output backup dump tape. When it does, it will request another

tape number from the operator. The reply to this question could have been issued in advance, by a call to "reply" in system_start_up.ec but the daemon will then take this string and call the tape software to have a tape mounted. Our suggestion so far has been to refuse to mount the tape.

It will not be difficult to change to the tape-assignment package to make it work differently for the Backup daemon if the system is unattended. When the operator is placing the system into unattended mode, he will load all available drives with blank backup tapes, and instruct the tape-assigment software assign these tapes one by one to the daemon as needed. This requires that the tape mount software perform a slightly different action when assigning a tape, that the tape DIM not dump a tape at mount time if it is already mounted, and also that the tapes which are not at load point when Multics is booted should be (file marked and) unloaded.

CONFIGURATION

The bug which prevents the system from coming up with more than one CPU should be fixed, so that a large system which crashes and successfully reboots is not limited to one CPU. There will be cases, however, when the supervisor deconfigures some system resources due to errors. (This is done with bulk store records currently, and will be done with memory in the future. With the new Storage system, disk drives may also be treated this way.) If a resource has been dropped due to error, an automatic restart should know not to try to use the device again.

GENERALIZATIONS OF THIS SCHEME

Unattended mode should probably be implemented as a set of switches, one for automatic rebooting, another for tape-drive handling, and so forth. Some installations may wish to provide weekend service with no operator in the computer room but with an attended operator console distant from the machine room (via the message coordinator). There are also devices on the market which automatically find a tape and load it onto a tape drive in response to a software order; so that an unattended system might be able to answer most tape mount requests. Another possibility which might be used by some sites is that one or more IO devices (card readers, tape drives, printers) might be located in an area accessible to trusted users, who would be able to issue commands to attach and use these devices even when no operator is attending the main computer.

The unattended modes should probably be implemented so that, when a problem occurs, it will be convenient for someone to come in, fix the problem, and let the system continue unattended. For

example, if a printer runs out of paper, and a system programmer happens to pass the machine, he should be able to insert a fresh box of paper and allow the daemon to continue printing, without having to issue any Initializer commands, or having to affect unatteded mode with respect to tapes.