To: Distribution

From: Bill Silver

Subject: Operator's Console Recovery

Date: 05/01/74

Introduction

The operator's console is used to type syserr messages. It is extremely undesirable for these messages to be lost. This document describes a mechanism which will allow syserr messages to be seen by the operator even though the console is inoperative for writing. In addition, this document will discuss how this mechanism allows the operator's console to be detached from the Initializer process for use by an on-line T&D process.

Basic Strategy

The basic strategy involves using the syserr logging and message coordinator mechanisms. Without both of these facilities, no console recovery is possible. The following scenario will follow a syserr message through the system and out onto a terminal. This is only a brief description. Full details will be given later in this document.

1. syserr_real is called by the hardcore procedure which generates the syserr message. The message is expanded by formline_ and put into the wired syserr log buffer. It is assigned a unique syserr sequence number. A special syserr log interrupt is generated.

2. syserr_logger is called to handle the syserr log interrupt. It copies the whole wired log buffer into its stack. It then copies all the messages that were in the wired syserr log buffer into the paged syserr log partition. The data associated with each message is also copied into the paged log.

3. ocdcm_ is called by syserr_real to write the message on the console. If, after retrying a number of times, ocdcm_ determines that the console is inoperative for writing, it will send a wakeup to the System Control process over a prearranged event call channel. The event message will be the sequence number of this syserr message.
4. The message coordinator in the system control process receives this wakeup. It will call ohcs_$ring_0_peek to obtain a copy of the part of the log segment that it

needs. Looking through this log, it will find the syserr message whose sequence number was specified in the event message.

5. This message will be copied from the log (really the work copy of the log). It will then be put into the message coordinator device queue associated with the terminal which was previously specified for console recovery.

6. Another wakeup is sent over an event call channel associated with this terminal. When the message coordinator receives this wakeup, it will take the message out of the device queue and output it on the terminal.

Retry and Recovery of Console Operations

ocdcm_ performs three types of services:

1. It reads messages from the console for ocdim_.
2. It writes messages on the console for ocdim_.
3. It writes messages on the console for syserr_real.

Errors can occur when performing any of these operations. The retry strategy for all three is the same. It is simple. Each operation which results in an error (errors due to some apparent console hardware problem) will be retried 10 number of times. After 10 straight errors, ocdcm_ declares that the operation has failed. It then goes into its recovery procedure. What it does depends upon whether or not all the necessary pieces of the recovery mechanism are available. It also depends upon which type of operation failed.

There are two requirements for the console recovery mechanism:

1. The syserr logging mechanism must be enabled. Each time ocdcm_ is called to write a syserr message, it is also passed the sequence number of that message. It always knows the sequence number of the last syserr message. If this syserr sequence number is zero, then the syserr logging mechanism is not enabled.

2. The message coordinator must be prepared to receive a wakeup from ocdcm_. When it is prepared to do this, it will create an event call channel for this purpose. It will call ocdcm_ to give it the ID of this event channel. If this channel ID is zero, ocdcm_ knows that the message coordinator is not ready to accept this wakeup.

Now we can discuss the recovery actions taken for each possible case:

1. Reading: This case is independent of whether or not the recovery mechanism is available. ocdcm_ will stop trying to read. It will lock the console. It will not try to read again until the user hits the REQUEST button. No wakeup is sent to the message coordinator.

2. Writing without the recovery mechanism: This case includes writes of both syserr and DIM messages. The action taken is to delete the message from its respective write buffer. If the syserr write buffer becomes full, subsequent calls to ocdcm_ will loop, waiting for space to become available. If ocdcm_ loops long enough, the system will crash. Thus, even at the expense of losing a syserr message, we must free space in the syserr write buffer. The case for DIM messages is similar, but not so critical. If the DIM write buffer is full, ocdim_ will be told to block. Unless we free space in the DIM write buffer, the System Control process will remain blocked in ocdim_. Note that losing a DIM message is not as harmful as losing a syserr message.

3. Writing DIM messages with recovery: The whole console recovery mechanism is designed to save syserr messages. Even if the recovery mechanism is available, the action taken when writing a DIM message fails, is the same as described above. In effect, there is no recovery for DIM messages.

4. Writing syserr messages with recovery: All the messages queued in the syserr and DIM write buffers will be deleted. The console will be declared to be in the recovery mode. A wakeup will be sent over the recovery event call channel set up by the message coordinator. The event message sent will be the sequence number of the syserr message that failed to be written.

Once ocdcm_ declares the console to be in the recovery mode, it does not stop functioning. syserr_real and ocdim_ will continue to call it. It will continue trying to read. It will continue to queue both types of messages and try to write them. When these operations fail, the actions described above will be taken.

The Message Coordinator's Part in Console Recovery

When the message coordinator is initialized, it will set up some data indicating that the console recovery mechanism is not enabled. In order to enable the mechanism, the message

coordinator needs control over a terminal type device. As soon as such a device is given to the message coordinator, it will enable the recovery mechanism. It will create an event call channel. It will initialize its current syserr sequence number to zero. This implies that although the recovery mechanism is available, as far as the message coordinator is concerned, it is not yet being used. A call will be made to ocdcm_ to give it the ID of the recovery event call channel. Until ocdcm_ receives this call it will not try to recover.

The message coordinator must maintain a pointer to the control block associated with the terminal it is going to use for recovery. Whenever a new device is given to the message coordinator, if the recovery mechanism is not already enabled, it will check to see if this device is a terminal. Each time a device is taken away from the message coordinator, if the recovery mechanism is enabled, it will check that this device is not the recovery terminal. If it is, the message coordinator will try to find another terminal to use for recovery. If it cannot find another, it will call ocdcm_ with a zero event channel ID. This tells ocdcm_ that recovery is no longer possible.

A special procedure in the message coordinator will receive the wakeups over the recovery event call channel. If the current syserr sequence number maintained by the message coordinator is zero, it knows that it is being called to begin using the recovery mechanism. It will perform necessary initialization. If the sequence number contained in the event message is negative, it knows that it is being called to stop using the recovery mechanism.

The general procedure followed each time that a recovery wakeup is received is as follows: A call to phcs_$ring_0_peek will be made to copy the part of the paged syserr log partition that is needed. Then, using the current sequence number and the sequence number sent in the event message, all the syserr messages that have not yet been written will be copied into the device queue associated with the recovery terminal. The message coordinator will then go about its normal way of writting messages that are queued for a particular device. Note that the message coordinator will check the syserr code of all the messages it takes out of the syserr log. Some syserr messages are only logged and not written. The message coordinator does not want to write these messages.

DETACHING THE CONSOLE FROM THE INITIALIZER PROCESS

Two new entry points will be added to ocdcm_ for use by on-line T&D. These entry points are "detach" and "reattach". These entry points may be called only through a new hardcore gate to which only an on-line T&D process has access. The function of

these two entry points is described below.

detach

This entry is called by on-line T$D when it wants to run tests on the console. ocdcm_ will wait for any active console I/O operation to complete. ocdcm_ will call iom_manager to unassign the console. It will declare the console to be in the detached mode. If the console can be placed in the recovery mode, it will be. In any case, both write buffers will be reset and console will keyboard will be locked.

ocdcm_ cannot initiate any I/O operations to the console when it is in the detached mode. However, it will still be called by ocdim_ and syserr_real. Calls from ocdim_, both read and write requests, will simply be ignored. Calls from syserr_real will be ignored if the console is not in the recovery mode. If the console is in the recovery mode, a wakeup will be sent over the recovery event channel. In either case the syserr message will not be queued. syserr_real will think that the message has been written. However, this will only be true if the recovery mechanism was enabled.

reattach

This entry point is called by on-line T&D when it has completed it console testing. At this time, the console should have been detached by the T&D process and should be available to the Initializer process. ocdcm_ will call iom_manager to reattach the console. The console will be taken out of the detached mode. If the console was in the recovery mode, it will be taken out of this mode.