MULTICS OPERATING STAFF NOTE

MOSN 6.4.3
Revision 0

To: Distribution

From: Robert S. Coren

Subject: OPERATION OF THE I/O DAEMON

Date: 1/1/74

This document describes the necessary procedures to run the I/O Daemon from the operational point of view. Other relevant documents are MOSN 6.4.3.1, "Messages Produced by the I/O Daemon," and MOSN 6.4.3.2, "Maintenance of the I/O Daemon."

I. PROCESSES REQUIRED FOR I/O DAEMON

The I/O Daemon subsystem consists of a single central process called the Coordinator, and one "device driver" (or "driver") process for each output device to be run.

II. BRINGING UP THE I/O DAEMON

A. Bringing Up the Coordinator

The Coordinator should be logged in before any drivers are brought up. I/O daemon processes have a login id of "IO", so in order to log in the Coordinator, type:

```
login IO
```

The system will reply:

```
Password:
```

and the operator must type the password associated with the I/O daemon.

The daemon will ask:

```
coordinator, driver, remote, or cards?
```

To indicate that this is the Coordinator, reply:

```
coordinator
```

and, as long as no Coordinator is already logged in, the Coordinator process will begin to run. If there is already a Coordinator, this response is invalid, and a message will be printed asking for a different response.

(Note: The response "cards" is only used for a card-reading daemon. See MOSN 8.3.1, "Card Input.")

B. Bringing Up a Driver

A device driver must be logged in for each device to be run. As for the Coordinator, log in a device driver by typing:

```
login IO
```

and when the system replies:

```
Password:
```

type the password for the I/O daemon. When the daemon asks:

```
coordinator, driver, remote, or cards?
```

the operator should reply

```
driver
```

indicating that the process is a device driver. The driver will then type:

```
Enter device name and optional device class:
```

The operator should reply with the channel identifier of the device to be run, followed optionally by a blank and the "device class" (i.e., the generic name for the type of device) to which the device belongs. If the device class is omitted, a default will be supplied. For example, the operator could type:

```
prta printer
```

or simply:

```
prta
```

in order to run the printer on channel "prta".

After validating the device id and attaching the device, the driver will type:

```
prta ready to start
```

and printing will proceed, assuming that any outstanding print requests are queued. In normal circumstances, no further operator intervention should be required to run the printer.

Additional processes may be logged in to run additional devices if desired, either from other terminals or a single terminal using the services of the Message Coordinator. To log in a second printer, for example, type once again:

```
login IO
```

and after the password has been entered and validated, the reply "driver" has been entered, and the new driver has asked for a device name:

```
prtb printer
```

whereupon the driver will start operating the printer on the "prtb" channel.

To start a process to run a card punch, log in a driver as above, and when asked for the device name, type:

```
puna
```

or:

```
puna punch
```

and the punch should be ready to operate.

C. Bringing Up a Driver for a Remote Device

Certain classes of devices are designated as "remote", i.e., the device is probably not in the same room as the central computer, and must communicate with it along a telephone line or the like. For such a device, the driver is initially logged in from a console like any other driver, but will subsequently listen for response on the line to the device.

For example, the operator types:

```
login IO
```

As usual, the system asks for a password and then types:

```
coordinator, driver, remote, or cards?
```

The correct reply in this case is:

```
remote
```

to indicate a driver for a remote device. The driver will then type:

```
          Enter device name:
```

The operator enters a name associated with the particular remote device, e.g.:

```
          mohawk
```

the driver process will now wait for the device's communication line to be dialled up. When the remote device dials in, another password may or may not be required (see MOSN 6.4.3.2, "Maintenance of the I/O Daemon," for more details), after which the driver will perform all output on, and expect all input from, the remote device.

Some remote devices may be run by processes on projects other than "SysDaemon", in which case the project name must be specified in the "login" command, for example:

```
          login IO Multics
```

Once the device has been dialled up and its password validated, the driver will say (at the remote device):

```
          Enter command.
```

the command may be any of the following:

```
          begin print
```

This command instructs the driver to begin processing print requests on the remote device.

```
          begin punch
```

If the device includes a card punch, the driver is to begin processing punch requests.

```
          begin print punch
```

Again assuming that the device includes both a printer and a punch, both print and punch requests, if available, should be processed, but print requests are to be given priority.

```
          begin punch print
```

This command is similar to the above, except that punch requests are to be given priority.

read

If the remote device is equipped with a card reader, an input deck is about to be fed into it. See MOSN 8.3.1, "Card Input."

D. Problems in Bringing Up the Daemon

If the device name or device class name is incorrectly specified, the driver will issue a message describing the error, such as:

```
Device class is invalid.
Device id is invalid.
No default device class for DEVICE
Device already running.
```

followed by the message:

```
Please try again.
```

The driver will ask for the device name and class again, and the operator should re-input them correctly.

In the case of more serious errors, the description of the problem will be followed by the message:

```
Driver process cannot be initialized.
```

and the driver process will be logged out. Action to be taken under these circumstances depends on the type of error; for further information, consult MOSN 6.4.3.1, "Messages Produced by the I/O Daemon."

If important data bases are missing or damaged, the Coordinator itself may not be able to function properly, in which case the Coordinator will produce an error message and log out. The condition described by the message (see MOSN 6.4.3.1) will probably have to be corrected by one of the procedures described in MOSN 6.4.3.2, "Maintenance of the I/O Daemon."

III. NORMAL OPERATION

Once a driver has started running its device, it should proceed without requiring further intervention. Segments (files) are printed or punched in response to user requests; each request is assigned an identification number composed of a sequence number and the number of the priority queue in which the request was placed. (Each driver uses its own set of sequence numbers.) Messages describing what is being printed or punched are written into a log, which may optionally be directed to the driver's console.

Example - typical messages in response to user requests:

```
Request 85.2: Printing >udd>m>JDoe>new_command.list for JDoe.Multics.a

Request  7.3: Punching >udd>SysDaemon>test_deck for Backup.SysDaemon.z
```

IV. I/O DAEMON COMMANDS

There are several commands which can be input to an I/O Daemon driver process in order to repeat, restart, or terminate requests. Most of these commands can be typed on the driver's input console at any time without interfering with output in progress; if a file is in the middle of being printed or punched, the command will not be processed until after the request is completed. In order to interrupt a driver in the middle of a request, hit the QUIT button on its input terminal, and after it types:

```
Enter command.
```

input whatever command is desired.

For the purpose of several of these commands it should be understood that the I/O Daemon saves sufficient information about each completed request to enable the request to be repeated later. This information is normally retained for a fixed amount of time, after which the request cannot be restarted. The time interval is installation-dependent and can be changed as described in MOSN 6.4.3.2; a probable value would be on the order of one hour.

The following are valid commands accepted by an I/O Daemon driver process.

attach

The "attach" command is used to re-attach a device that has been detached by a "detach" command (see below); the "attach" command is only valid if a "detach" command has been entered previously. To enter the "attach" command, type:

```
attach
```

The "reattach" command is equivalent to the "attach" command.

cancel

The "cancel" command causes the request currently being executed to be aborted and not saved for later restarting. It might be used if a bad file was causing troubles on the printer and ought not to be repeated in case of a "restart" command (see below); it might also be used at the request of a user who inadvertently requested something he didn't want.

To enter the "cancel" command, hit the QUIT button, and type:

```
cancel
```

In order to make sure that the correct request is cancelled, the driver will type:

```
Do you wish to cancel request n?
```

where n is the sequence number of the current request. If the operator replies "yes", the request will be cancelled, and the driver will proceed to the next request; if the reply is "no", the request will be restarted from the beginning.

The "cancel" request is invalid if not preceded by a QUIT, or if entered when no request is active; in either case the driver will print the message:

```
No current request.
Enter command.
```

and await further input. The most usual operator reply in this situation is the "start" command.

detach

The "detach" command is used to detach a device temporarily (in case it needs manual attention, for instance) without destroying the process that is running it. To enter the "detach" command, type:

```
detach
```

After detaching the device, the driver will type the message:

```
Enter command.
```

and wait for further input. The only commands that are valid while the device is detached are "attach" (or "reattach"), "reinit", and "logout".

If the "detach" command is entered when output has been suspended in response to a QUIT, the current request will be aborted (see the "kill" command).

kill

The "kill" command behaves similarly to the "cancel" command, except that the current request is saved for possible later restarting.

To enter the "kill" command, hit the QUIT button, and type:

```
kill
```

As with "cancel", the driver will ask:

```
Do you wish to kill request n?
```

and if the answer is "no", it will restart the request instead.

The "kill" command is invalid if not preceded by a QUIT, or if no request is active.

logout

The "logout" command is used to cause the driver process to be logged out. It would be used when it was desired to shut off the device, or run the device attached to another system, or start running the device as a member of a different device class.

To enter the "logout" command, type:

```
logout
```

The driver process will log out after it has processed the current request, if any. If the "logout" command is preceded by a QUIT while output is in progress, the current request is killed and saved (as if in response to a "kill" command), and then the driver logs out.

reattach

The "reattach" command is simply another way of entering the "attach" command (see above).

reinit

The "reinit" command is used to reinitialize a driver process; it causes the process to detach its device, make itself known to the Coordinator as a new driver, and reattach the same device, without requiring the device id to be input again. It might be used if the process somehow got itself into an inconsistent state.

To enter the "reinit" command, type:

```
reinit
```

Unless the command is preceded by a QUIT, the current request will be completed before the process is reinitialized; if the "reinit" command is typed after a QUIT in the middle of an output request, the request will be aborted as if in response to the "kill" command, and will have to be restarted after the process is reinitialized.

restart

The "restart" command can be used to serve either of two functions: 1) to restart the current request from the beginning (e.g. if the printer ran out of paper in the middle of a request); or 2) to repeat all requests previously performed by a particular device, starting from a specified identification number (e.g., if a printer has been doing bad things for some time, like overprinting the same line again and again).

To perform function 1), hit the QUIT button and type:

```
restart
```

If the driver was in the middle of a request, it will respond:

Request _n_ restarted.

where _n_ is the sequence number of the current request. It will then restart request _n_ from the beginning. If the driver was between requests, or if the command was not preceded by a QUIT, it will type:

```
No current request.  Enter command.
```

and await further commands.

For function 2), a QUIT is not necessary. Simply type:

```
          restart n DEVICE
```

where n is the sequence number of request from which restarting is to begin, and DEVICE is the name of the device whose requests are to be restarted. DEVICE may be omitted, in which case the name of the device being run by the driver to which the command was typed is used.

The effect of this command is to repeat all requests performed by DEVICE, starting with request n (if it is still available). If more than one device of the same class as DEVICE is running, all drivers of that class will work on repeating DEVICE's requests until all the desired requests have been redone.

A couple of examples may help to clarify the use of this command. Assume that two printers are being run, one with a device name of "prta" and the other with a device name of "prtb". Let us say that "prta" is performing a print request with a sequence number of 25, and "prtb" is doing one with an sequence number of 16. (Remember that each device has its own set of sequence numbers.) The operator types at the input console for "prta":

```
          restart 13
```

indicating that all of "prta"'s requests from 13 on are to be reprinted. As soon as they have finished their current requests, both "prta" and "prtb" will start working on "prta"'s requests 13 through 25 (without duplicating each other's work). As soon as one or the other of the printers has completed the repetition of request 25, both of them will resume processing of the user request queues, "prta" numbering its next new request as 26, and "prtb" numbering its next request as 17 (since it had interrupted its regular work after request 16).

As a second example, suppose that printer "prta" has developed some kind of mechanical trouble which has caused all of its requests from number 13 on to be bad, and further that this trouble is not immediately fixable, so that "prta" has to be detached or logged out. The operator may then type at the input console for "prtb":

```
          restart 13 prta
```

Although "prta" is disabled, "prtb" will repeat all those requests previously performed by "prta" from 13 up through 25 (the last request); then it will resume processing requests from the queue.

It may happen that, when a "restart" request is issued, the information describing the specified initial request has already been deleted, because the time limit for saving such information has been passed. This situation will be detected by the Coordinator, which will print the message:

```
Request n for device DEVICE is gone.
```

Assuming that _any_ of the device's requests are still available, the Coordinator will apply the "restart" command to the earliest one it can find, appending the following message:

```
Restarting from request m.
```

where _m_ is the oldest of DEVICE's requests that is still available. If none of DEVICE's requests are still available, the Coordinator will report:

```
No saved requests for device DEVICE
```

To avoid this situation when a device breaks and is going to take some time to fix, use the "save" command described below.

Note that if a device id is specified in the "restart" command, the specified device must be of the same class as the device to whose driver the command was addressed. For example, the command:

```
restart 13 prta
```

addressed to a driver for a card punch would be invalid, and would produce the following message from the Coordinator:

```
prta is not of the same class as puna
```

and no action will be taken.

save

The "save" command prevents completed requests from being lost when the time limit for normal restarting runs out. It might be used if a number of requests need to be restarted, but the printer has to be fixed first, and it is feared that the requests will be lost before the "restart" command can be issued.

To enter the "save" command, type:

```
save n DEVICE
```

where n and DEVICE have the same meanings as for the "restart" command above. As a result of this command, no requests performed by DEVICE from request n on will be deleted until after they have been redone in response to a "restart" command. A "save" command should always be followed eventually by a "restart" command specifying the same request number and device name, to prevent saving the requests for the duration of the session.

As with the "restart" command, it may be that some of the desired requests have already been lost. In this case, the Coordinator will warn:

```
Request n for device DEVICE is gone.
Saving from request m
```

or:

```
No saved requests for device DEVICE
```

start

The "start" command is used to resume processing after a QUIT. In addition, if an invalid command is entered, or a "kill", "cancel", or "restart" command is entered when no output is in progress, the driver will type:

```
Enter command.
```

and the normal processing of requests can be resumed by entering the "start" command.

To enter the "start" command, type:

```
start
```

Additional commands for remote devices

The following commands have meaning only for a remote device.

begin

The "begin" command is used to start processing certain types of requests, depending on its arguments. The "begin" command is entered as follows:

```
        begin TYPE1 TYPE2
```

where TYPE1 and TYPE2 are each either "print" or "punch" (but they may not be both the same). If only TYPE1 is entered, then requests of TYPE1 will be processed, and requests of the other type will not. If TYPE1 and TYPE2 are both present, both print and punch requests will be processed, but requests of type TYPE1 will be given priority. At least TYPE1 must be present.

read

The "read" command is used to request the device to suspend both printing and punching in order to read a deck input to the card reader. After the deck has been read, the device will await a further command.

The format of the "read" command is:

```
        read
```

stop

The "stop" command is used to discontinue processing one type of request. Its format is:

```
        stop TYPE
```

where TYPE is either "print" or "punch". If TYPE is "print", print requests will no longer be processed, but if punch requests were running previously, they will continue to be processed. The converse applies if TYPE is "punch". If as a result of a "stop" command, neither print nor punch requests are being processed, the driver will produce the following message:

```
        Print and punch are both inactive.
        Please enter fresh request.
```

and await another command (most likely a "begin" command, although it could also be "read" or "logout").

Input to the Coordinator

It is almost never necessary for the operator to send any input to the Coordinator process directly. The only way to do so is to send a QUIT to the Coordinator, after which one of the commands described below may be entered. It is not advisable to leave the Coordinator idle in a QUIT state for an extended period if there are active driver processes.

logout

The "logout" command causes the Coordinator to be logged out. This will stop all I/O Daemon activity; driver processes (if any are logged in) will reinitialize themselves and wait for a new Coordinator to be logged in.

The "logout" command is entered by typing:

```
logout
```

start

The "start" command causes the Coordinator to resume whatever it was doing before a QUIT. It might be used on a heavily-loaded system if the Coordinator is slow to come up (though this use is not recommended); it should be used after the "term" command described below.

To enter the "start" command, type:

```
start
```

term

The "term" command (for "terminate") is used in the event that, owing to some serious error condition, a driver process has terminated abnormally without being able to notify the Coordinator. The Coordinator must therefore be notified manually that the driver no longer exists; otherwise, no new process will be able to run that driver's device. The "term" command is used to inform the Coordinator that there is no longer a driver for a specified device.

To enter the "term" command, type:

```
term DEVICE
```

where DEVICE is the device name associated with the driver being terminated. When the Coordinator has cleared the specified driver from its data bases, it will say:

```
Enter command.
```

The normal response is the "start" command.

V. PROBLEMS AND RECOVERY

A. Minor Problems

If the following message appears on the driver's console:

```
*****PRINTER OUT OF PAPER*****
```

the operator should service the printer, after which the daemon will proceed from the point of interruption.

If the printer runs away or prints garbage, the chances are it is the fault of a bad user request, which can be aborted by hitting the QUIT button, and entering the "kill" or "cancel" command as described in the previous section. If possible, consult the submitter of the request before entering the "cancel" command; he may be able to rescue the request for later restarting (e.g., by changing the name of the file being printed).

B. Unclaimed Signals in the Driver Process

A message on a driver's console that begins:

```
iodd_signal_handler_:
```

followed by an error message signifies a program fault in the driver. If the error occurs during the processing of an output request, and the condition described is "segment fault" or "incorrect access on entry" when attempting to reference the file to be printed or punched, the message may be ignored. Any other message should be saved and reported to the programming staff. If an error recurs repeatedly, try entering the "reinit" command described in Section IV.

In most cases, after such an error message, the I/O Daemon will resume operation by proceeding to the next request. In certain rare cases, the driver process will be forced to reinitialize itself. Only if the process is apparently in a loop (i.e., the same message appears over and over) is operator intervention required; in such a case, hit the QUIT button and enter the "reinit" command.

C. Unclaimed Signals in the Coordinator Process

A message on the Coordinator's console beginning:

    io_Coordinator: Error:

and continuing with an error message indicates a fault in the Coordinator process. This is most likely the result of a program error, and should be called to the attention of a systems programmer. The I/O Daemon will generally recover and continue processing without intervention; however, if the same message appears over and over, it may be necessary to send a QUIT to the Coordinator and log it out.

More information on the subject of error messages from both driver and Coordinator processes may be found in MOSN 6.4.3.1, "Messages Produced by the I/O Daemon."