Published: 01/23/69

Identification

output_driver: The Output Driver Daemon
J. F. Ossanna

Purpose

This section describes a module that manages delayed (queued) printing and punching. Commands that result in such queued output requests include dprint, dpunch, and qdump7.

General

The output driver can operate in any(one) standard process. Once initialized, the output driver is entirely event-call driven, and other commands can be issued in the same process. Following every output completion, a comment is written on user_output specifying the channel name, the original name of the segment whose copy was just outputted, and an output identification number. The output id is for subsequent use by an operator in requesting reprinting or repunching. Commands exist for: (1) reoutputting segments when additional copies are needed or when the first copy is damaged; (2) changing the length of the circular list of outputted segments used for reoutputting; (3) adding to or deleting from the peripheral complement being controlled by the driver; and (4) listing the peripheral complement along with the busy state of each peripheral.

Initialization

The following command is used to add to the peripheral complement known to the driver.

output_driver$init type name

type is the segment name of the Device Interface Module (DIM) which operates a particular kind of device. type is either "pr202" (printer DIM) or "pun21" (punch DIM.) name is a valid channel name such as "prtb40". The first call to output_driver$init causes initialization of the driver's internal static and of the segment >peripherals_dir>daemon_events. The latter contains the driver's process id and the event channel names to be used to signal the driver.

Upon receipt of the init call, the driver issues an attach call directly to the indicated DIM, with the mode argument = "daemon". This mode causes the DIM to declare its hardware interrupt event channel to be an event call channel directed at one of its own entry points. The driver also includes as an additional last argument the per-instance pointer ordinarily supplied by the I/O switch.

Operation

Following initialization, operation of the output driver is automatic. The following command can be issued to obtain a list of the current peripheral complement.

output_driver$list

The driver will list the channel names known to the driver along with whether or not each peripheral is busy.

The following commands can be issued to reduce the peripheral complement.

ouptup_driver$detach name

or

output_driver$force_detach name

name is the same channel name used in the output_driver$init command. The first call will queue detachment of name until the corresponding peripheral is finished outputting the segment being currently outputted. The second call forces immediate detachment of name regardless of the busy state of the corresponding peripheral.

Following the completion of printing or punching of a segment, the driver writes onto user_output the following: (1) the channel name on which completion occurred; (2) the original name of the segment whose copy was just outputted; and (3) an output identification number. The following command can be issued to reoutput a segment.

output_driver$redo id

id is the output identification number written out at the time of original completion. id is an index into a circular list; any particular value of id will eventually

be reused for another segment. Thus reoutputting of a particular segment must be done before its id is reassigned. The size of the circular list can be altered by the following command.

output_driver$set_lmax n

n is the desired list size and can be as large as 30. This call with n missing can be used to have the current list size written onto user_output.

Implementation Overview

Actual queueing of output requests is done by the Interprocess Communication Facility. When all the peripherals of a given type become busy, the event channel corresponding to that type is changed to an event-wait channel. Later, when a peripheral becomes free, the event channel is changed to an event-call channel to enable receipt of event calls at the entry point "output_driver$work". The event id received with the event call is a unique bit string corresponding to the unique name of a control segment in >peripherals_dir>. The control segment contains information about the output request, including the pathname of the segment to be output.

The output of a segment is initiated by a single write call to the DIM. The DIM returns immediately after physically initiating the output of the beginning of the segment. The output of the remainder of the segment is achieved by the DIM on receipt of event calls at the time of each hardware interrupt. When the segment is completely output, the DIM calls output_driver$done to indicate the completion; the per_instance pointer associated with each instance of the DIM identifies which output completion occurred.

Upon receipt of this completion indication, the driver writes onto user_output the completion message previously described, and enters information into a circular list to permit the operator to request reoutputting.