Published: 01/05/68

Identification

Overview of Operator Commands
K. J. Martin

Purpose

The operator or operators of a Multics system have a set of commands available to them alone. These commands enable the system operator to give directives to the system, and any other operators to interact with the system in performing their duties. This section itemizes the facilities necessary for an operator to act effectively and the restraints which must be placed on operators.

The Operator's Environment

Operators see a Multics essentially identical to that seen by ordinary users, although operators will generally use a different set of commands and utility procedures than other users. Most of these procedures are not even available to the ordinary user. This difference in outlooks, however, is on the same level as the difference in outlooks of a business programmer and an artificial intelligence researcher. That is, the same Multics supports them all; they merely find that different sets of techniques and tools are pertinent to their problems.

Should it become necessary at an installation to restrict the operators' field of action, the restriction is standard and is described in BQ.2.05.

The system operator's interface to the system is outlined in BQ.1.01 on the System Control procedure. In brief, System Control serves as a dispatcher of commands from the system operator. One of System Control's responsibilities is to keep track of who is currently logged in as the system operator. The system operator helps keep track of any other operators by appropriately delegating authority.

The inter-process communication facility (BQ.6) is heavily used in relaying information between operators and other parts of the system. In general, the operator commands place information in a data base or remove information from it, perhaps do some processing, then send an event to the appropriate process. When the user requests an action by the operator, his request is relayed to the operator by signalling an event.

The operators at an installation are assigned functions to perform. These functions are distinguishable duties and are each associated with either a system process or with some system module. The set of functions assigned to an operator may vary as the needs of the installation vary. Two types of functions exist: 1) those where all inter-process communication is initiated by the operator, and 2) those where another process-group initiates communication. The latter are referred to in these documents as unsolicited-request functions. A prime example of an unsolicited-request function is media management where users issue directives to operators regarding media handling. These functions pose special problems (discussed later) since the appropriate operator's working process id must always be available to them.

In Initial Multics two operator functions are defined: system and media. The system function is the type 1 above and involves tasks such as specifying communication line configuration. The operator's process communicates with System Control in the performance of the system function, and with the Media Request Management module (see BT.2) in the performance of the media function.

Special Facilities

Users working on the sys_operator project (that is, the operators) do have a few special facilities and requirements. First, each operator must be identified with the function(s) he is to perform. When Multics is initialized and the first operator logs in, his login responder calls op_report (BX.15.02) to indicate to System Control that he is available for assignment. This first operator is required to log in from one of a number of specified consoles and to log in on the project "system-operator". Requiring that he log in on the project "system-operator" guarantees that only a bona fide operator can be system operator; further restricting him to log in on a specified console (on a specific GIOC line) further guarantees that he is at an appropriate location. When the first operator logs in who satisfies these requirements, System Control assumes that he is to be the system operator and will perform all operator functions until notified of a change. System Control records what information it needs (see BQ.1.01). It also records in the operator's process-group directory (in the segment, op_function) a list of functions that this operator performs. When subsequent operators log in, they also call op_report (automatically via their login responders) and wait for duties to be assigned.

When the system operator wishes to assign some of his duties to another operator (or in any way revise the distribution of duties) he executes the delegate command (BX.15.05). The system operator is the only one allowed to delegate. System Control updates its own records and the records in the individual operators' process-group directories, then signals the operator to inform him of the function newly assigned to him. The effect of this updating is that each operator has a segment op_function in his process-group directory which contains a list of the functions he is to perform.

The segment op_function is used to implement a sit-and-wait mode for the operators who are in charge of unsolicited-request functions such as media management. Every operator is given a special login responder (see BQ.2.05 for a description of login responders and quit responders). This login responder is simply a special version of the Listener (BX.2.02). It differs from the standard Listener in that after reading a command sequence it appends the op_checker command (BX.15.03) to that command sequence. The composite command sequence is then passed to the Shell. The effect in the media operator's case is that after executing the other commands in the sequence the Shell calls op_checker which waits for media requests, and calls media command (BX.15.09) when one occurs. The operator can get out of op_checker only by quitting. (More about quits below.)

The effect of all this is that the media operator is forced to pay attention to media requests following each command sequence he issues. It is thought that requiring explicit action <u>not to see</u> the requests is better than requiring explicit action <u>to see</u> the requests. The media operator may explicitly invoke the media command if he wishes.

Quits are a special problem with operators who are responsible for unsolicited-request functions. When an operator first takes over a function, the op_here procedure (BX.15.02) is executed for him. Op_here informs the appropriate system module(s) of the operator's working process id and an event channel over which the system module may send events. If the operator quits he destroys that working process, leaving an invalid process id with the system modules. The solution is to provide all operators with a special quit responder which re-exercises the op_here procedure after each quit. Thus the id of the operators' new working process is given to the appropriate system module(s).

The special login and quit responders required for operators are described in BX.15.01. In Initial Multics neither will be available. Thus the media operator will be required to check for possible media requests in a reliable manner. Each operator will also be required to issue the op_report command upon logging in and to issue the op_here command following each quit.

Operator Command Standards

The operator commands are required to be modular in nature. This is necessary because an installation may have any number of operators and any combination of responsibilities. Any two distinct tasks should be divided into two separate commands.

Operator commands should be the clearest and simplest of all to use, because these commands are issued with relatively high frequencies, by users who, in general, are not programmers. Because of this, an operator is less prepared to cope with a poorly designed command.