Operating systems for small computers are more restricted in their capabilities than those used on larger computers and are generally not designed for multiprogramming. However, some facility is needed to permit such actions as inquiries.

This paper discusses an option to a small operating system that permits interruptions of a job, execution of a program to handle an inquiry, and return to the interrupted job. A description of the operating system serves as a basis in discussing the characteristics of the option.

On-line inquiry under a small-system operating system

by K. Darga

One of the most inconvenient restrictions to the user of a single-partition-oriented operating system with no multiprogramming capabilities is not having the computer available for other work during execution of a large program. Many users, however, want facilities that allow the execution of certain programs at any time. In particular, they are interested in putting urgent inquiries to their data files on disk, without being forced to abort the job in hand. For systems with limited main storage, a roll-in/roll-out function, which frees main storage for the execution of an intermediate program, can be an adequate solution to this problem.

Such a function is offered by the IBM System/360 Model 20 Disk Programming System (DPS) and is called an inquiry function.[1] DPS was designed specifically for a computer system having a small main storage and limited capabilities. In this paper, a description of the DPS is first given as background for the discussion of the development of the inquiry function, and is followed by the general concept of the inquiry function. Of particular interest are the various resource protection problems that had to be solved for the system to function properly. It is shown that the protection feature is the key to the operation of the inquiry function.

The Model 20 disk operating system

The Disk Programming System is the disk-oriented operating system for the Model 20. Programming systems that were made for the larger models of the System/360, such as the Operating System or Disk Operating System, cannot be used for the Model 20 because of several differences in hardware facilities. To contribute to a proper understanding of Model 20 programming support, some of the distinguishing characteristics of the Model 20 are listed:[2,3] (1) There is no distinction between the supervisor state and problem state, and consequently no supervisor-call instruction. (2) There is no hardware protection feature and no wait state. (3) Interruptions do not include machine and program checks; instead, the checks lead to hard machine stops. (4) In the input/output operations, there is no channel address word.[4,5] Unit record devices are not started by channel command words, but by a special XIO-instruction that contains all information necessary to start an input/output unit itself. The instruction set of the Model 20 is basically a subset of the instructions of the larger models but has, in addition, some instructions that are incompatible with the larger models.

DPS monitor

Designed as a low-cost computer, this machine did not include a multiprogramming capability, mainly because of the first two characteristics listed. Consequently, DPS was developed as a single-partition programming system.[6] During execution, main storage is occupied by that part of the control program for the system that resides there, called the *monitor*, and one problem program. The basic routines of the monitor are (1) the program loader which fetches load modules (phases) from the core-image library (section of a storage device in which program elements have been assigned the actual main storage addresses they will occupy) into main storage, (2) the end-of-job routine which fetches the job control program into storage to initiate the next job, (3) the input/output routine which starts input/output requests for disk and tape, (4) interruption-handling routines, and (5) the input/output scheduler which queues those input/output requests that cannot be started immediately.

Additional parts of the resident control program are a communication area and the physical and logical unit blocks. The communication area is divided into two parts. One of them, the permanent link data area (PLDA), contains several switches, pointers, and entry addresses reserved for the exclusive use of the computer system. The other part of the communication area is provided for user programs. It contains information such as the actual date, a user program switch that can be set by job control cards, and areas available for intraprogram and interprogram communication.

Normally, the size of the monitor does not exceed 4,600 bytes. Its size depends on several options that can be requested at monitor-generation time. For example, one of these options makes parts

of the monitor transient; that is, these parts do not remain in main storage throughout the system operation and are overlaid by other called routines. The inquiry support is another such option.

Monitor services for the problem program are requested by direct branches into fixed locations of the permanent link data area. From here, control is given to the individual service routines, such as the disk input/output start routine. The monitor can be considered as a pool of system resources available for the problem program.

system disk pack

Another pool of system resources is the system disk pack (SYSRES) which contains the system as one data file. This file is divided into two extents (devices and track boundaries), one containing the loading program, the other containing the system libraries and their directories. The library extent consists of the following areas: (1) a system directory that describes the location of the individual libraries and directories within this file, (2) one section that contains the monitor in load module format, (3) the core-image library and its directory that contain all programs to be executed in load module or pointers to each module, (4) the macro facility library and its directory that contain macro definitions in an internal format or pointers to each macroinstruction (macro definitions can only be used in assembler programs; the internal format is understood by the assembler), and (5) the relocatable area, used as a work area for compile-and-go or assemble-and-execute runs (the Report Program Generator (RPG) compiler and the Assembler put the generated object program into the relocatable area; the user can instruct the system to include the contents of the relocatable area in the core-image library as a temporary entry and then immediately execute the program).

A special data file located on the system disk pack is the label information area (LIA), which serves as a communication area between data management routines (IOCS) and the job control function. All pertinent file information, which must be provided by means of job control cards, is put into this area and can be retrieved from this area by the IOCS routines of the user's program.

Another characteristic of DPS is that it is a batch-oriented system. That part of main storage not occupied by the monitor is alternately used by a problem program and the job control program. The function of job control is to select control information required for the next job to be executed and to cause transfer of control to the first phase of this job. This job could be a user-written program or one of the system programs. The DPS programs include:

- The programming language translators: Assembler and RPG compiler
- Several utility programs such as disk sort, tape sort, and file-to-file utilities

- Library service programs which are used to add or delete modules to or from the libraries

In addition, DPS provides a set of macro definitions that can be used in assembler programs to generate IOCS routines, routines that require monitor services, and a monitor.

The inquiry support

The inquiry option of DPS allows stacked-job processing to be interrupted for the execution of an intervening program, called an inquiry program. To initiate an inquiry program, the user presses the REQUEST key on the printer-keyboard causing an interruption of the job in progress that indicates to the system the user's intentions to start an inquiry program. The system checks to see whether or not the current mainline program has indicated any objections against the execution of inquiry programs. An objection would result, for example, if the mainline program turned on an inquiry control bit in the communication region. If the system does not encounter this indicator, it asks for the name of the inquiry program the user wants executed. This is accomplished by issuing the message ENTER PROGNAME on the printer-keyboard. In accordance with the instructions of this message, the system sets up a printer-keyboard read instruction, giving the operator the opportunity to enter the name of the inquiry program. When the name has been entered, the system saves the status of the mainline program by writing it into a reserved area of the core-image library. Then the inquiry program is fetched from the same library and control is transferred to it. After execution of the inquiry program, the status of the interrupted mainline program is restored, and the latter resumes execution.

Observe that the execution of inquiry programs is not preceded by a job control function. Label information and assignments of input/output devices required for the execution of a specific inquiry program can be provided by any previous job control run in the batched-job stream and is stored as permanent information by the system.

input area

An interesting feature of the DPS inquiry support is the printer-keyboard input area, which can be incorporated into the monitor at its generation. The implementation of this feature was determined by the expected structure of application inquiry programs. Such programs select one or more records from an indexed sequential file and display these records or parts of them via the printer-keyboard. The decision as to which records are to be displayed occurs at execution time. The input information required for this decision is provided by the operator, for example, by typing the keys of the records on the printer-keybroad. If this typing takes place at execution time of the inquiry program, it probably cannot be overlapped meaningfully with other processing. The inquiry program spends

most of the time waiting for information, and the interrupted mainline program is held up for an inconvenient length of time.

The incorporation of a printer-keyboard input area into the monitor allows its use as a communication area. The key information for the standard inquiry program can be typed when the name of the inquiry program is typed. The system automatically sets up a second printer-keybroad read instruction at inquiry-initiation time thus allowing the operator to enter information into the input area. The inquiry program can retrieve this information later. The advantage of using the input area is that the operator's typing can be completely overlapped with processing of the mainline program.

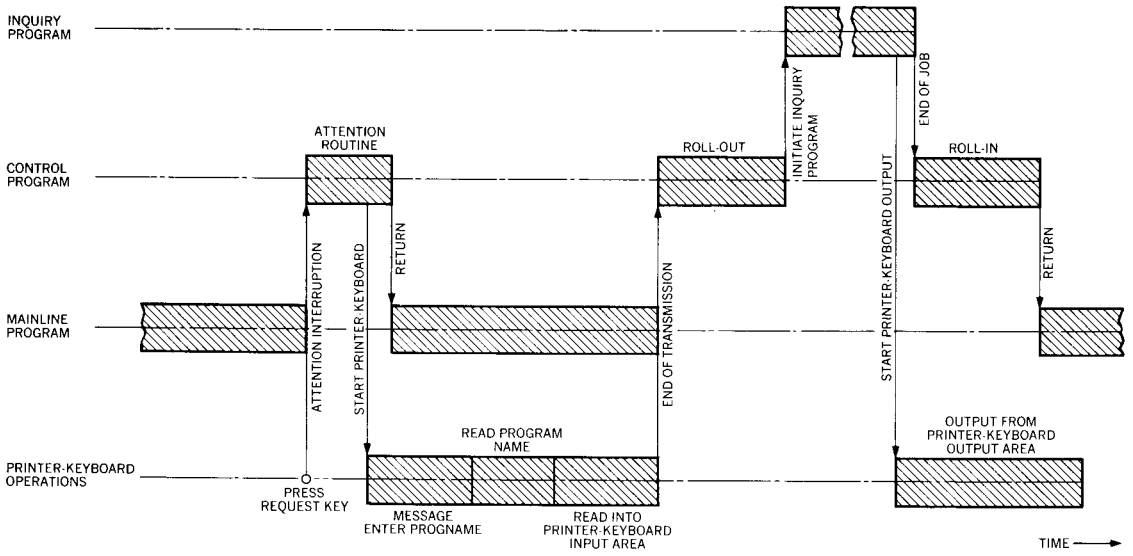
output area

Another feature available in the DPS inquiry support is the printer-keyboard output area, which can be incorporated adjacent to the monitor when the monitor is generated. Its purpose is similar to that of the printer-keyboard input area. If the output area is used in an inquiry program, the output at the end of the program can be overlapped with the restoring and processing of the mainline program, provided that the interrupted mainline program does not make use of, or overlay, the printer-keyboard output area.

The simplified schematic flow of events that takes place after a request for execution of an inquiry program is illustrated in Figure 1. This figure also shows all operations on the printer-keyboard being overlapped with processing of the current mainline program.

Although the inquiry facilities were made for application programs requiring the operations described, the user cannot be prevented

Figure 1 Execution of an inquiry program



from calling any program residing in the core-image library. For that reason, the system must have safeguards. Protection methods to prevent unauthorized access to resources are discussed in the following section.

Protection of system resources

The intermediate execution of a program during the processing of another one implies problems similar to those that exist when two or more programs are executed concurrently in multiprogramming. Difficulties are apt to arise when the intervening program attempts to access any serially reusable system resources that are currently in use by the interrupted mainline program.[7] Such system resources could be input/output devices, data files, libraries, or routines of the control program in main storage.

access prevention

From experience with multiprogramming systems, we know the following two general methods to prevent simultaneous access to serially reusable resources:

1. Enqueuing requests for such resources, which means that any program requesting a serially reusable resource and finding it allocated to another program has to wait (is not dispatched) until the resource is available.
2. Preventing those programs that need currently occupied resources for their execution from entering main storage.

For a roll-in/roll-out function, the first method is not applicable. A program allocated to a serially reusable resource could not release this resource if it were rolled out of main storage.

The second method can only be employed when the system knows in advance all resources that are required for the execution of a program. This would force the user to provide control information for all resources he uses in his program, so that a program initiator could check for resource conflicts with the program currently in process. DPS automatically handles all control of serially reusable system resources without requesting any control information to be provided by the user. The following discussion is a description of the protection features implemented to prevent uncontrolled errors.

Resources in DPS that are the object of protection can be grouped into three classes: (1) monitor routines, (2) the system file on the system disk pack, and (3) user data files.

monitor routines

Routines of the monitor usually run disabled (all interruptions masked off in the program status word), so that any request for inquiry service is delayed until the routines are finished. If the request occurs during execution of a monitor routine when inter-

ruptions are temporarily not masked off for some reason (for example, during program fetch), the inquiry service is logically delayed to prevent simultaneous access to serially reusable monitor resources (transient area). Monitor routines are normally not very time-consuming; the delay of the inquiry program is not very great.

system file

Various protection methods are implemented for the libraries within the system file. When a system service program is in the process of rearranging the core-image library or the core-image directory, or when another service program is reorganizing the whole system file, it would not be possible to fetch any inquiry program from the core-image library. The system is informed of this situation by an inquiry control bit in the communication area. If the operator were to try to start an inquiry program, he would get a message on the printer-keyboard indicating that no inquiry programs can be executed for the duration of the current mainline program.

Another type of resource conflict could occur when a system service program running in the batched-job stream is rearranging, for example, the macro facility library. This library is only required for the execution of a very few system programs. No user-written program would get access to this library.

It would not be reasonable to prevent the execution of inquiry programs in general when the library is in a pending state. For this reason, indicators are set in the communication region by programs that temporarily disarrange those areas of the system file not necessarily required for the execution or control of inquiry programs. System programs that require access to such areas of the system file test these indicators when they run as inquiry programs. If they find an indicator on, they are aborted and processing of the interrupted mainline program is resumed.

user data files

One of the main problems in a roll-in/roll-out environment is that of protecting shared direct-access storage-device files against simultaneously being updated. Let us briefly illustrate this problem.

Assume that the mainline program is updating the records of a disk file. A record may be read into main storage and updated. Before it is written back onto a disk, an inquiry program is started. But this inquiry program updates the same record of that specific file. When the mainline program later resumes execution, it writes its update to the same disk position, and the results of the updating by the inquiry program are completely lost. If the inquiry program, instead of updating a record, added records to the same disk file which the mainline program is processing, the file could be destroyed.

The easiest way to prevent these problems from occurring would be to eliminate any sharing of data files. However, this restriction would not be justified. If both the mainline program and the inquiry

program only read records of the same disk file, no difficulties would occur.

Before the file protection mechanism of DPS is explained, let us briefly consider the IBM System/360 Operating System (OS/360).[8] A multiprogramming system such as OS/360 has exactly the same problems of providing protection of shared files. OS/360 allows concurrent access to the same file (data set) by more than one program if the user declares in a control statement which file is to be shared. Any control over serializing of updating operations on a shared data set is left to the user's responsibility. The system only helps by providing the necessary supervisor service with use of the macroinstructions ENQ and DEQ.[9] The user has to request explicitly in his program the exclusive control over a data set or records of a data set. The system serializes any subsequent requests to the same resources by enqueuing them. For the OS/360 enque and deque service to work properly, all programs must follow the conventions and ask for the resources before using them.

file protection

File protection in DPS is controlled by data management routines. The OPEN routine for any disk file in a mainline program sets protection bits in the label of that file. These protection bits indicate which specific operations are currently performed on the file (for example, updating or additions, etc.).

The OPEN routine of any inquiry program automatically tests the protection bits. The protection bits set by the mainline program may cause the inquiry program to be aborted if it attempts to perform certain operations on the protected file. If, for example, the mainline program is in the process of updating a data file, any attempt of an inquiry program to update the same file or to add records to the file would be prohibited. An inquiry program, however, that only retrieves records from the same file would be allowed to run. The conditions under which indexed sequential disk files are protected from being accessed by inquiry programs and those under which such access is allowed are shown in Table 1.

Table 1 Protection of indexed sequential files

	Inquiry program			
Mainline program	*LOAD extend*	*UPDATE*	*ADD*	*RETRIEVE*
LOAD extend	P	P	P	P
UPDATE	P	P	P	A
ADD	P	A	P	A
RETRIEVE	P	A	P	A

P—Access to file is not allowed for inquiry program
A—Access to file is allowed for inquiry program

The protection bits in the file labels are cleared in the CLOSE routine of the mainline program. File protection for direct access files and for sequential disk files is achieved in the same way as that for indexed sequential files.

Performance considerations

Functions available in a programming system are usually evaluated according to two main features: the main storage requirements and the execution time requirements. Therefore, it might be of interest to know how much main storage is required in the monitor for the inquiry function, and how much time it takes to initiate an inquiry program.

The main storage requirements of the inquiry function within the monitor are rather small, since roll-in and roll-out are transient routines. The number of bytes by which the monitor increases is about 300, excluding the printer-keyboard input and output areas. The time which is required for the initiation of an inquiry program depends on the size of main storage. For a machine with a main storage of 16K bytes, it takes about one second. This time, of course, does not include the time needed by the operator to type the program name or any inquiry program input. But as the operator's typing can be overlapped by processing of the mainline program, this time is of no importance.

Summary

The original purpose of the DPS inquiry support—to provide a function that allows inquiries to disk files during stacked-job processing—was surpassed. The DPS inquiry function more generally offers the possibility of starting almost any type of high-priority program during stacked-job processing without aborting the current job in the job stack. This function is offered in a small system with reasonable safeguards to protect the user against mistakes.

CITED REFERENCES AND FOOTNOTES

1. Some concepts similar to those of the inquiry function were available in the IBM 1401 Disk System and the IBM 305 RAMAC.
2. G. M. Amdahl, G. A. Blaauw, and F. P. Brooks, Jr., "Architecture of the IBM System/360," *IBM Journal of Research and Development* **8,** No. 2, 87–101 (April 1964).
3. G. A. Blaauw and F. P. Brooks, Jr., "The Structure of System/360, Part I—Outline of the logical structure," *IBM Systems Journal* **3,** No. 2, 119–135 (1964).
4. *IBM System/360 Model 20 DPS Input/Output Control System,* Systems Reference Library, C24-9007, International Business Machines Corporation, Data Processing Division, White Plains, New York.
5. A. Padegs, "The structure of System/360, Part IV—Channel design considerations," *IBM Systems Journal* **3,** No. 2, 165–180 (1964).

6. Documentation pertaining to the IBM System/360 Model 20 Disk Programming System published by the International Business Machines Corporation, Data Processing Division, White Plains, New York, includes: *Guide to DPS* (C33-6001), *System Generation and Maintenance* (C33-6006), *Operating Procedures* (C33-6004), and *Performance Estimates* (C33-6003).
7. J. W. Havender, "Avoiding deadlock in multitasking systems," *IBM Systems Journal* **7,** No. 2, 74–84 (1968).
8. The concepts of the IBM System/360 Operating System are described in more detail in the three-part article "The functional structure of OS/360," *IBM Systems Journal* **5,** No. 1, 2–51 (1966).
9. *IBM System/360 Operating System Supervisor and Data Management Services,* Systems Reference Library, C28-6646, International Business Machines Corporation, Data Processing Division, White Plains, New York.